Fast Simulated Diffusion: An Optimization Algorithm for Multiminimum Problems and Its Application to MOSFET Model Parameter Extraction

Takayasu Sakurai, *Member, IEEE*, Bill Lin, *Member, IEEE*, and A. Richard Newton, *Fellow, IEEE*

***Abstract*—A new algorithm, namely a fast simulated diffusion (FSD), is proposed to solve a multiminimal optimization problem on multidimensional continuous space. The algorithm performs a greedy search and a random search alternately and can find the global minimum with a practical success rate. A new, efficient hill-decending method employed as the greedy search in the FSD is proposed. When the FSD is applied to a set of standard test functions, it shows an order of magnitude faster speed than the conventional simulated diffusion. Some of the optimization problems encountered in system and VLSI designs are classified into multioptimal problems. A MOSFET parameter extraction problem is one of them and the proposed FSD is successfully applied to the problem with a deep submicron MOSFET.**

I. Introduction

SOME of the VLSI design problems including transistor sizing and model parameter extraction can be regarded as a minimization problem in multidimensional continuous space with an object function which has plural local minima. Well-established minimization procedures for convex functions, for example, the Levenberg–Marquardt method [1], can be easily trapped in one of the local minima and thus cannot find a global minimizer. Recently a method called simulated diffusion (SD) has been proposed [2] for finding the global minimum of a multiminimal function on continuous space. The simulated diffusion is conceived by the stimulus of simulated annealing (SA), which is for combinatorial optimization problems [3]. Although much effort has been made to theoretically study the behavior of the SD [4], [5] and it has been demonstrated theoretically that under certain conditions the method will find the global minimizer with a probability of unity, little is known about the practical aspects of the SD as an optimization procedure. Although the SD can find a global minimizer, it is very slow [2].

In this paper, a new optimization method, called fast simulated diffusion (FSD), is proposed to provide a faster way of finding the global minimum. The new method is successfully applied to MOSFET parameter extraction problems in the deep submicron regime.

In Section II, the basic idea of the conventional SD is described. In Section III, the algorithm of the fast SD is presented, and the advantage of the fast SD over the conventional SD is clarified in Section IV. Section V is dedicated to a discussion on the application of the proposed fast SD method to the practical VLSI design problems, namely a MOSFET model parameter extraction problem for a circuit simulator. The results are summarized in Section VI.

Manuscript received July 9, 1990; revised January 29, 1991. This work was supported by a grant from the Toshiba Corporation. This paper was recommended by Associate Editor J. G. Fossum.

T. Sakurai is with the Semiconductor Device Engineering Laboratory, Toshiba Corporation, Kawasaki, 210 Japan.

B. Lin and A. R. Newton are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 9104220.

II. Conventional Simulated Diffusion (CSD)

First, the basic idea of conventional simulated diffusion is described. Essentially, SD makes use of the physical fact that a particle placed in a given potential and with Brownian motion is diffused into the global minimum of the given potential profile. The following is a more mathematical formulation of the process. A differential equation which describes a diffusion process of a particle with Brownian motion is given as

$$dx = -\nabla f(x)\,dt + \sqrt{2T}\,dw \tag{1}$$

where t is time, x is the space coordinate which indicates the location where the particle is, $f(x)$ is a potential function in which the particle is put, ∇ is a gradient operation, dw is Gaussian random noise, and T is temperature. The first term on the right side corresponds to the drift component of the movement and the second term signifies the Brownian movement. When the temperature is high, the second term dominates and the movement of the particle is just stochastic. On the other hand, when the temperature becomes low, the first term dominates and the process approaches pure hill descent. The second term is essential to get out of the local minimum and the first term gives the tendency to minimize the function.

It has been shown [4] that with a proper cooling schedule, the probability distribution of x, $P(x)$, approaches

$$P(x) \propto \exp\{-f(x)/T\} \tag{2}$$

as t goes infinity. This means that the limit distribution is independent of the initial value and is peaked around the global minimizers of $f(x)$. This in turn means that if dx is

integrated over a long period of time, x tends to converge to a global minimum of the function $f(x)$. This is the principle of conventional simulated diffusion. Aluffi-Pentini *et al.* [2] numerically integrated (1) to obtain the minimizer from this first principle. However, the numerical process turned out to be slow.

If there are constraints in the original minimization problem, it is possible to introduce penalization functions and make it a minimization problem without constraints [6]. Consequently, SD can be applied not only to unconstrained minimization problems but also to optimization problems with constraints.

III. Fast Simulated Diffusion

In this work, instead of directly integrating (1), two basic modifications are made. One is the introduction of an accept/nonaccept function of a Boltzman distribution type, which is commonly used in simulated annealing. If the next point, x_{next} $(= x + dx)$ gives a smaller function value than the current x, take x_{next}. On the other hand, if x_{next} gives a larger function value than the current x, generate a random number R in [0, 1] and calculate $P = \exp[-\{f(x + dx) - f(x)\}/T]$. If $R < P$, then accept x_{next}; otherwise discard the x_{next} and regenerate x_{next}. The higher the function value becomes in the next move, the less probable it becomes to accept the move. This point selection rule was first introduced in [13] and was shown to be effective in simulated annealing in [2].

The introduction of this Boltzman accept/nonaccept function into the simulated diffusion algorithm can be validated by (2), which is the Boltzmann distribution itself and it is expected to help establish the probability distribution of (2) faster than simply integrating (1). In practice, the use of this accept/nonaccept function prunes very "stupid" moves that would otherwise be made and consequently accelerates the convergence.

The other modification concerns the generation of the next move. Instead of adding the greedy hill-descending part (the first term of (1)) and the random perturbation part (the second term of (1)), the generation of x based on a greedy method and a random method is carried out *alternately*. That is, dx is first calculated by $-\nabla f(x)\,dt$ and is then calculated as $\sqrt{2T}\,dw$. By generating the next move by the gradient method and the random method alternately, it is possible to achieve hill descent even if the temperature is relatively high. In the relatively high temperature range, the random term happens to generate ineffective moves and it is probable that no improvements of $f(x)$ will be observed if the two terms are added together as in the CSD. This is because the hill-descending part can be hidden by the dominating random noise and all moves are possibly rejected. In the above description, the term *greedy* means that the method rejects any unprofitable move, although the drawback is temporary and in a long run the move is profitable.

Several considerations other than the above-mentioned two major modifications make the method more efficient. First, since it is expensive to calculate the direction of $\nabla f(x)$ if the space has large dimensions, $<\nabla f(x) \cdot r > r$ is used instead, where r is a unit vector of a randomly picked axis. This is because the expected direction of $<\nabla f(x) \cdot r > r$ approaches $\nabla f(x)$ in the long run [2]. Second, since it is difficult to choose a good value of dt, a new hill-descending method is proposed and used. The choice of dt is critical because if it is too small, the improvement of the solution is small, but if it is too big, $-\nabla f(x)\,dt$ does not always give an improvement. The proposed method is described in Fig. 1. First, pick a random axis direction. If the function is concave at the point along the picked axis, quadratic fitting is carried out and the minimum x in that direction is guessed and adopted as x_{next}. If the function is convex, choose a small dx first and double the dx until $f(x + dx)$ fails to decrease from $f(x)$. The doubling process is confined up to a certain number of times (three in the following examples). It is not an objective of this new hill-descending method to give the exact minimum in that direction but rather to provide an inexpensive yet effective way of improving the solution, since there is always a possibility that the random search can give rise to a big jump, in which case the previous hill-descending becomes wasteful. This method is regarded as an inexpensive adaptive method for determining a good value of dt. One may argue that more expensive, but efficient, algorithm could be employed near the global minimum. However, in practice it is difficult to know if the current point is near the global minimum and a big jump may take place after the expensive improving steps and all the efforts become vain.

A rough sketch of the FSD algorithm is shown in Fig. 2 and a detailed description of the FSD method is shown in Fig. 3. In the first several external loops (around ten loops), hill descending is not taken and only random search is carried out because big jumps are accepted in the high-temperature stage and hill descending is not effective at all.

The initialization scheme and the temperature update algorithms in [7] are adopted. That is, the initial temperature, T_{init}, is determined by statistics gathered over randomly selected N_{init} points, as shown in Fig. 3. The adopted temperature update algorithm (cooling schedule) is basically a geometric decrease. The theory of SD suggests that the cooling schedule should be much slower than the geometric decrease to guarantee that the global minimum will be reached even for ill-conditioned functions [11]. However, for practical problems, geometric cooling works well [7], [12]. Although T_{init} and N_{init} affect the performance of the FSD, they are not as critical as the cooling schedule.

The initial distribution of dw is chosen so that almost all the feasible space is covered by the random search at the initial stage. Such a distribution can be determined when the feasible region of x is given as a supercube, $[x_{\min}, x_{\max}]$. In practical problems, this feasible region is known in advance (see Section V) or is set sufficiently large. If the randomly generated x falls out of the feasible

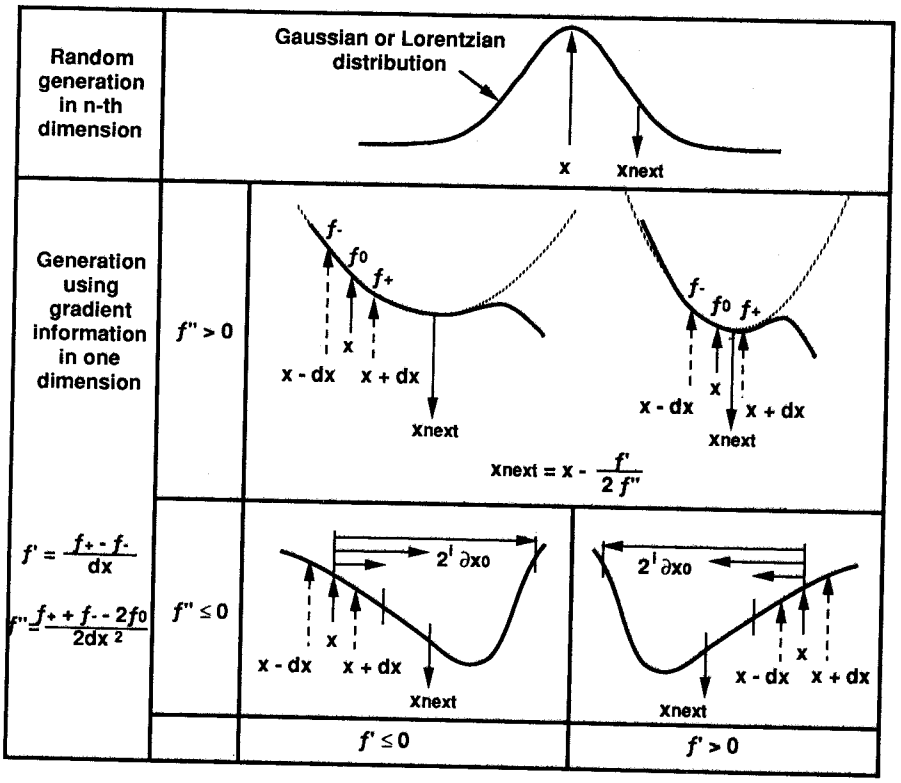


Fig. 1. Proposed hill-descending method using f' and f'' information.

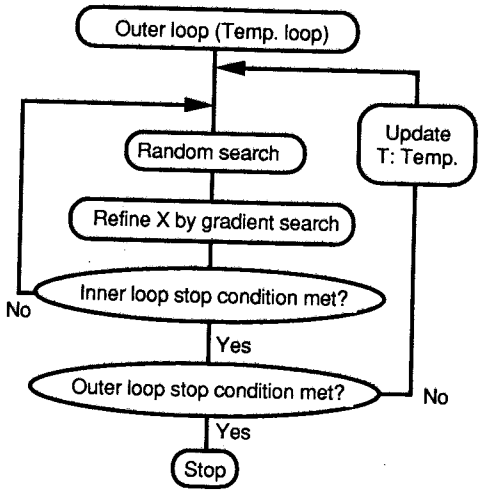


Fig. 2. Rough sketch of the fast simulated diffusion algorithm.

region, it is regenerated. At the last stage of FSD, when the object function shows little change, the Last_Gasp sequence is taken, whereby the temperature is increased a little and then decreased to freeze. The details are shown in Fig. 3.

In Fig. 3, a multiplier, S, controls the random search space volume. S should be shrunk in proportion to $\sqrt{T}$ as T is lowered according to the first principle of SD, but in practice S can be reduced faster and is proportional to T_n ($n = 0.5 \sim 1.0$).

IV. Comparison Between FSD and CSD

Table I shows a comparison between FSD and CSD when they are applied to a set of standard test functions given in [2]. On the average, FSD is about an order of magnitude faster than CSD. Let us define *reachability* as the probability of finding the global minimum in a finite period of time using the given algorithm. The number of successful trials in ten trials in Table I can be used as an index of reachability. Improvement in efficiency or speed might be obtained at the risk of degrading the reachability. Judging from Table I, the reachability of FSD is in the practical range.

When the first term in (1) is neglected, the method becomes similar to SA. This SA-like method is thought to be better than the mere extension of SA to a continuous space [8], since the random search space is decreased by a factor of $\sqrt{2T}$ as the temperature is lowered. FSD is faster than this SA-like method, as shown Table II, because fewer "stupid" moves are generated. In the stochastic search method, it is important to prune the search space, and FSD provides a way of pruning the search space effectively.

In Table II, the results of using a Lorentzian distribution [9] instead of a Gaussian distribution are also shown. Further improvement in both speed and reachability is observed. Since the Lorentzian distribution has a longer tail than the Gaussian distribution, the possibility of a large jump is rather high with the Lorentzian distribution even

```
Algorithm of Fast Simulated Diffsion

main {
    T = Tinit = (κ * (standard deviation of f(X) over randomly selected Ninit points));
                                    // Set initial temperature by using huristics. κ=0.2, Ninit=200
    S = Sinit;                      // Set initial S to Sinit
    Xinit = (Xinit_Given_by_User or one of those randomely selected Ninit points
            whichever gives the minimum value of f);
    Xopt = X = Xinit;               // Set initial X to the best X known
    do {                            // External loop with varying T
        iINT = 0;
        while (a certain times (ex. 15~25*dimension)) {// Internal loop with constant T
            iINT ++;
            Generate_X();           // Generate new X by simulated diffusion
            Δf= f(Xnew) - f(X);
            If (Δf < 0) {           // If cost decreases,
                X = Xnew;           // adopt the Xnew.
                If ( f(X) < f(Xopt) ) { Xopt = X } // Save best X.
            } else {                // Even if cost increases,
                P = exp(-Δf / T);   // adopt the Xnew
                R = random number in [0,1]; // according to
                If (R < P) {X = Xnew}       // Boltzman distribution.
            }
        }
        if ( f(X) > f(Xopt) ) { X = Xopt }      // Resume the best X.
        If (cost is not improved considerably) {
            iLast_Gasp ++;          // If cost is not improved considerably,
        } else {                    // take Last_Gasp loop,
            iLast_Gasp = 0;         // where T is increased a little
        }                           // and then decreased to freeze.
        Update_T();
        Update_S();                 // S = Sinit * (T / Tinit)^a ; a = 0.5~1(ex. a=0.75)
    } until ((iLast_Gasp > iLast_Gasp_Max) and (T / Tinit < T_Ratio_Min) )
                // until Last_Gasp loop is taken long enough and T gets low enough.
    solution = Xopt;
}

Generate_X() {                      // generate new X
    If (iINT < mINT) {
        gradient_Flag = 0;
    } else {
        gradient_Flag = 1 - gradient_Flag;
    }
    If (gradient_Flag == 1) {
        Randomly select single variable Xi and move only in this axis.
        Generate Xnew with gradient information according to f" and f' values.
    } else {
        Xnew = X + S * (n-th dimensional Gaussian or Lorentzian distribution)
    }
}

Update_T() {                        // update temperature
    If (iLast_Gasp = 0) {
        T_Factor = exp(-λ T / σ)    // ex. λ=0.7, σ=standard dev. of accepted f(x)
        If (T_Factor < T_Factor_Min) { T_Factor = T_Factor_Min (=0.5)}
        T *= T_Factor
    }
    If ( 1 ≤ iLast_Gasp ≤ n2)   { T *= T_Factor2 (T_Factor2 > 1, ex. 1.3) } // ex. n2=4
    If (n2 ≤ iLast_Gasp)            { T *= T_Factor1 (T_Factor1 < 1, ex. 0.75) }
}
```

Fig. 3. Detailed algorithm of fast simulated diffusion.

at low temperature and it helps to get out of the local minima at the final stage.

V. Application to MOSFET Model Parameter Extraction

The model parameter extraction problem is to minimize the object function

$$f(p) = \sum_{\text{various bias conditions}} \text{weight (bias condition)} \cdot /I_{D,\text{measured}} - I_{D,\text{model}}(p)/ \quad (3)$$

with the model parameters, p, as variables. In the above expression, I_D denotes the drain current of a MOSFET and the weight function is optional. The SPICE LEVEL3 MOS model is used as a MOS model in this section as an example, although the method is not restricted to specific device models. The model parameters, p, that minimize $f(p)$ are considered to form a good extracted parameter set and can be used for the circuit simulation afterwards. With the conventional extraction program, the extracted parameters give the local minimum of $f(p)$ which is the closest to the given initial parameter set [1]. However, in practice, it is difficult and often impossible to guess the initial parameter set correctly. FSD does not require an initial value. All information needed beforehand is on the bounds, $p_{\min}$ and $p_{\max}$, for each parameter. This is rather easy because it is known that, for example, the parameter

TABLE I
COMPARISON OF CONVENTIONAL SD AND FAST SD

problem description			CSD (*1)	Fast Simulated Diffusion (this work)		
problem # (*1,2)	dimension	# of local minima	NF1: # of function evaluation	NF2: # of function evaluation (*3)	success rate in 10 trials (*4)	NF2/NF1 (%)
1	1	3	7168	3644	1.0	50.8
2	1	19	77699	2586	1.0	3.3
3	2	760	241215	3067	1.0	1.3
4	2	760	76894	2968	0.8	3.9
5	2	760	183819	2734	0.7	1.5
6	2	6	10822	4573	1.0	42.3
7	2	25	159549	3408	1.0	2.1
8	3	125	72851	3572	1.0	4.9
9	4	625	49690	3818	1.0	7.7
10	5	1e5	72226	5246	1.0	7.3
11	8	1e8	136061	9819	0.9	7.2
12	10	1e10	98985	12206	1.0	12.3
13	2	900	23770	4081	1.0	17.2
14	3	2.7e4	66010	4036	1.0	6.1
15	4	8.1e5	122166	4473	1.0	3.7
16	5	7.6e5	66365	4588	1.0	6.9
17	6	1.1e7	98974	5559	1.0	5.6
18	7	1.7e8	109886	6509	0.9	5.9
average	3.8	5.7e8	93009	4828	0.96	5.2

*1) See [2].
*2) Expressions for problems 4 and 5 in [2] seem to contain errors and so have been modified.
*3) Average over ten trials.
*4) This is the rate of having reached the global minimum in ten trials. This information is not contained in [2], which gives only "yes" or "no" in one trial as the reachability information.

TABLE II
TWO MODIFIED VERSIONS OF FAST SIMULATED DIFFUSION

		Simulated Annealing-like random search			Simulated Diffusion with Lorentzian Distribution		
problem #	NF1: # of function evaluation	NF3: # of function evaluation	success rate in 10 trials (Table I *4)	NF3 / NF1 (%)	NF4: # of function evaluation	success rate in 10 trials (Table I *4)	NF4 / NF1 (%)
1	7168	3111	1.0	43.4	2939	1.0	41.0
2	77699	3060	1.0	3.9	2387	1.0	3.1
3	241215	4131	0.7	1.7	2877	1.0	1.2
4	76894	5967	0.7	7.8	3170	0.8	4.1
5	183819	5831	0.7	3.2	2678	0.7	1.5
6	10822	5151	0.9	47.6	3609	1.0	33.4
7	159549	7701	0.9	4.8	3023	1.0	1.9
8	72851	11322	1.0	15.5	3232	1.0	4.4
9	49690	11475	1.0	23.1	3401	1.0	6.8
10	72226	20053	1.0	27.8	4108	1.0	5.7
11	136061	28689	0.9	21.1	7716	1.0	5.7
12	98985	33986	1.0	34.3	9856	1.0	10.0
13	23770	7378	1.0	31.0	3294	1.0	13.9
14	66010	10761	1.0	16.3	3446	1.0	5.2
15	122166	11424	1.0	9.4	4051	1.0	3.3
16	66365	14790	1.0	22.3	4140	1.0	6.2
17	98974	19730	1.0	19.9	4903	1.0	5.0
18	109886	22962	1.0	20.9	6295	1.0	5.7
average	93009	12640	0.93	13.6	4174	0.97	4.5

KAPPA is in the range of $0 \sim 2$. The values used for the bounds are tabulated in Table III. The same set of bounds is used to extract the 0.25 μm and 1 μm MOSFET parameters.

In order to further increase the efficiency in this particular problem of parameter extraction, the search is carried out in the logarithmic space for NSUB, VMAX, and NSS. This measure is taken to achieve a balanced search over a space because, for example, VMAX is in the range of $1e4 \sim 1e8$ and the increase from $1e4$ to $1.1e4$ tends to have an effect on $I_{D,\text{model}}$ similar to that of an increase from $1\ e7$ to $1.1e7$. For other parameters, the search is made on a linear scale.

TABLE III
MOSFET MODEL PARAMETER EXTRACTION RESULTS

parameter name	p_{min}	p_{max}	extracted params for 1μm MOS	extracted params for 0.25μm MOS
VTO	0	1.5	0.769	0.743
UO	10	1000	900	406
NSUB	1e16	1e20	1.80e17	5.97e18
GAMMA	0.2	1.5	0.928	0.477
ETA	0	2	0.0293	0.00754
THETA	0	2	0.996	0.775
KAPPA	0	2	0.382	0.299
VMAX	1e4	1e8	5.26e7	1.81e5
XJ	1e-8	3e-8	2e-7(fixed)	2.02e-8
TOX	-	-	2e-8(fixed)	5e-9(fixed)
NFS	-	-	0(fixed)	0(fixed)
LD	-	-	0.1(fixed)	0(fixed)
W	-	-	10e-6(fixed)	4e-6(fixed)
L	-	-	1.0e-6(fixed)	0.25e-6(fixed)
# of func. eval. time (min.•MIPS)	-	-	4258 ~18	3114 ~13

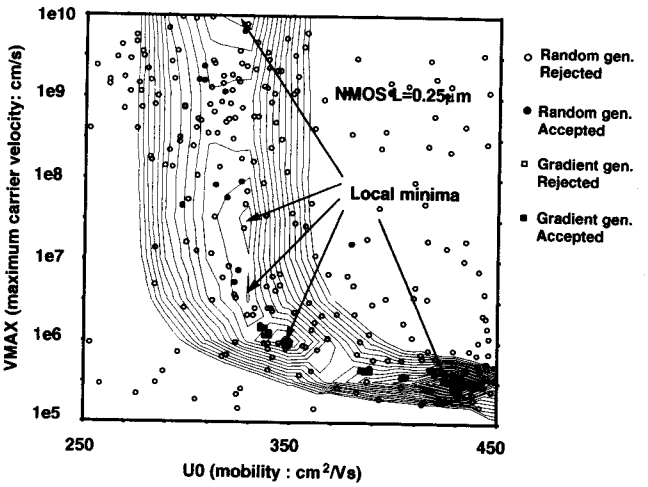


Fig. 4. Multiple-minimal nature of MOS model parameter extraction problem and generated x points.

The multiminimal nature of the object function is shown in Fig. 4 together with the generated x points with FSD. An example of the fitted drain current is shown in Fig. 5 for a 1 μm MOSFET. Fig. 6 shows another example of parameter extraction with a 0.25 μm channel length MOSFET [10]. Good agreement is observed even down to the deep submicron region. This indicates the effectiveness of the LEVEL 3 MOS model in the submicrometer region if the extracted model parameter set is used only for a narrow range of channel lengths. Usually the shortest channel length is used for almost all the MOSFET's in a VLSI and two or three sets of parameters are enough in designing a whole VLSI. Separate parameter sets are also required for a very narrow width device, a shallow V_{TH} device, and an i-type (intrinsic V_{TH}) device if they are employed. Even though it is good practice to use model parameters near the condition where they are

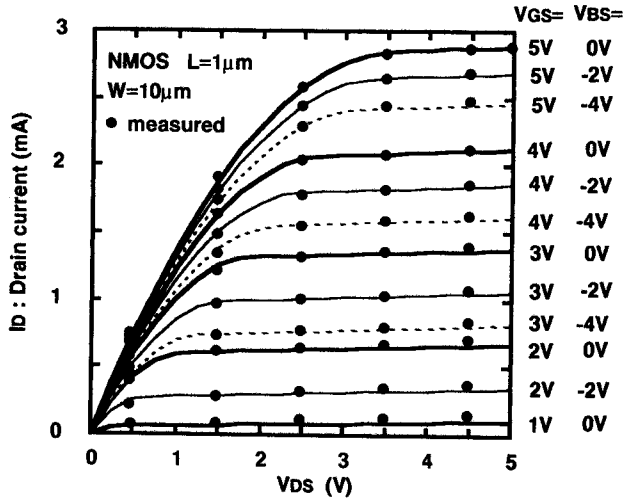


Fig. 5. Measured V_{DS}–I_D data (dots) for 1 μm MOSFET with SPICE LEVEL3 MOS model calculation (lines) fitted to them.

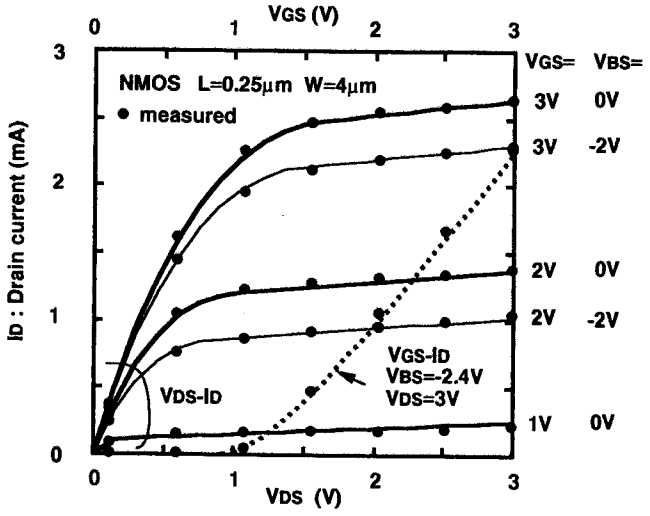


Fig. 6. Measured V_{DS}–I_D data (dots) for 0.25 μm MOSFET with SPICE LEVEL3 MOS model calculation (lines) fitted to them.

extracted, more advanced models, such as the BSIM, can cover a wider range of channel-length and threshold-voltage variations with a single parameter set.

VI. Conclusions

Fast simulated diffusion is proposed as a fast method for finding the global minimum of a multiminimal function on multidimensional continuous space. The technique is about an order of magnitude faster than conventional simulated diffusion when applied to a set of standard test functions. Fast simulated diffusion is successfully applied to MOSFET model parameter extraction in the deep submicron region. The method is believed to be applicable to other optimization problems encountered in system and VLSI designs [14]. The salient feature of the FSD is that it carries out an inexpensive solution refinement process after each stochastic search. In this sense, the method could be applicable to noncontinuous-space problems.

Acknowledgment

The encouragement of Prof. B. Brayton, Prof. A. Sangiovanni-Vincentelli, Y. Unno, Y. Takeishi, H. Yamada, and T. Iizuka throughout the course of this work is appreciated. Discussions with Prof. E. Polak and Dr. F. I. Romeo on general multiminimal optimization problems and simulated annealing were extremely useful. The assistance provided by H. Ishiuchi, T. Quarles, and R. Spickelmier concerning MOS physics, SPICE, and computer environments is also appreciated. The authors are grateful for the useful comments of the reviewers, which have substantially improved the paper.

References

[1] K. Doganis and D. L. Scharfetter, "General optimization and extraction of IC device model parameters," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1219–1228, Sept. 1983.

[2] F. Aluffi-Pentini, V. Parisi, and F. Zirilli, "Global optimization and stochastic differential equations," *J. Optimiz. Theory Appl.*, vol. 47, no. 1, pp. 1–16, Sept. 1985.

[3] S. Kirkparick, C. D. Gellatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[4] S. Geman and C-R Hwang, "Diffusions for global optimization," *SIAM J. Contr. Optimiz.*, vol. 24, no. 5, pp. 1031–1043, Sept. 1986.

[5] T-S Chiang, C-R Hwang, and S-J Sheu, "Diffusion for global optimization in R_n," *SIAM J. Contr. Optimiz.*, vol. 25, no. 3, pp. 737–753, May 1987.

[6] A. V. Fiacco and G. P. McCormick, "Computational algorithm for the sequential unconstrained minimization technique for nonlinear programming," *Management Sci.*, vol. 10, pp. 601–617, 1964.

[7] M. D. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing," in *Proc. ICCAD'86*, Nov. 1986, pp. 381–384.

[8] M. K. Vai and M. F. D. Ng, "A technology-independent device modeling program using simulated annealing optimization," in *Proc. CICC'89*, May 1989, pp. 9.4.1–4.

[9] H. Szu and R. Harley, "Fast simulated annealing," *Phys. Lett. A*, vol. 122, nos. 3,4, pp. 157–162, June 1987.

[10] M. C. Jeng, P. K. Ko, and C. Hu, "A deep-submicron MOSFET model for analog/digital circuit simulations," in *Proc. IEDM'88*, Dec. 1988, pp. 114–117.

[11] F. I. Romeo, "Simulated annealing: Theory and applications to layout problems," U. C. Berkeley ERL Memo, No. UCB/ERL M89/29, Mar. 1989.

[12] G. B. Sorkin, "Simulated annealing on fractals: Theoretical analysis and relevance for combinatorial optimization," in *Proc. 6th Ann. MIT Conf. Advanced Res. VLSI*, Apr. 1990.

[13] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *J. Chem. Phys.*, vol. 21, p. 1087, 1953.

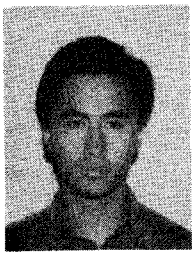
[14] T. Sakurai *et al.*, "Simple expressions for interconnection delay, coupling and crosstalk in VLSI's," in *Proc. ISCAS'91*, June 1991, pp. 2375–2378.

Takayasu Sakurai (S'77–M'78) was born in Tokyo, Japan, on **January 10, 1954. He** received the B.S., M.S., **and Ph.D. degrees** in electronic engineering **from the University** of Tokyo in 1976, 1978, **and 1981, respectively**. His Ph.D. work was **on the electronic structures** of the Si–SiO_2 interface.

In 1981 he joined the Semiconductor Device **Engineering Laboratory**, Toshiba Corporation, **Kawasaki, Japan**, where he did research **and development work** on CMOS dynamic RAM's **and**

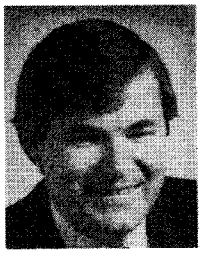
64 kbit, 256 kbit SRAM's, 1 Mbit virtual SRAM's, cache memories, a RISC with on-chip cache memory, and BiCMOS ASIC's. During the development, he also worked on the modeling of interconnect capacitance and delay, soft-error-free memory cells, new memory architectures, hot-carrier-resistant circuits, arbiter optimization, and gate-level delay modeling. From 1988 through 1990, he was a visiting scholar at the University of California, Berkeley, doing research on computer-aided design of VLSI's. He is currently back at Toshiba in charge of memory-embedded system VLSI development. His present interests include application-specific memories, BiCMOS VLSI's, VLSI microprocessors, and CAD for VLSI's.

Dr. Sakurai is a member of the Institute of Electronics, Information and Communication Engineers of Japan and the Japan Society of Applied Physics.



Bill Lin (S'87–M'90) received the B.Sc. degree in electrical engineering and computer sciences in 1985 from the University of California, Berkeley. Since 1985 he has been doing graduate work at Berkeley in electrical engineering, where he is currently a doctoral candidate specializing in the area of computer-aided design of VLSI integrated circuits. He has also worked with the Hughes Aircraft Company and the Hewlett Packard Company in the areas of parallel processing and software interfaces.

Mr. Lin received a best paper award at the 24th Design Automation Conference (DAC), in 1987, a distinguished paper award at the 1989 IFIP VLSI *Conference in Munich, Germany, and a distinguished paper citation at the* 1990 International Conference on Computer-Aided Design (ICCAD). His research interests include various aspects of CAD for VLSI, with emphasis on logic synthesis, verification, and asynchronous circuits. He is a member of Tau Beta Pi and Eta Kappa Nu.



A. Richard Newton (S'73–M'78–SM'86–F'88) is a Professor in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley, and served as vice-chairman of the department from 1984 to 1988. He has been actively involved as a researcher and teacher in the area of computer-aided design and computer architecture for 13 years. His special interests are synthesis (behavioral, logic, physical), design of integrated circuits, and multiprocessor implementation of algorithms.

He has consulted for many companies in the area of computer-aided design for integrated circuit design, including the Digital Equipment Corporation, General Electric, Hewlett-Packard, Intel, Synopsys, SDA Systems, Silicon Systems, Tektronix, and the Xerox Corporation. In addition, he is a member of the technical advisory boards of Sequent Computers, Candence Incorporated, and Objectivity.

Dr. Newton served as the Technical Program Chair of the 1988 and 1989 ACM/IEEE Design Automation Conferences. He was also an associate editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 1985 to 1988 and a member of the Circuits and Systems Society ADCOM. He has received a number of awards, including best paper awards at the European Solid State Circuits Conference and the 1987 ACM/IEEE Design Automation Conference, and he was selected in 1987 as the national recipient of the C. Holmes McDonald Outstanding Young Professor Award of the Eta-Kappa-Nu Engineering Honor Society. In addition, Dr. Newton supervises the research of *more than a dozen graduate students working in the area of computer-aided* design for VLSI systems.