Monte-Carlo Approach for Power Estimation in Sequential Circuits

Vikram Saxena†, Farid N. Najm and Ibrahim N. Hajj
ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign

Abstract

In this paper we present a Monte-Carlo based statistical techniques for estimating power in sequential circuits. Mutually independent samples of power are generated by simulating multiple copies of the circuit. Since the approach is simulation-based, spatiotemporal correlations are automatically accounted for. The algorithm iterates until the user-specified accuracy is achieved. Experimental results on ISCAS89 circuits show that reliable results can be obtained in considerably less time than that required by exhaustive simulation.

1. Introduction

The dramatic decrease in feature size and the corresponding increase in the number of devices on a chip, combined with the growing demand for portable communication and computing systems, have made power consumption one of the major concerns in VLSI circuits and systems design [1]. Excessive power dissipation in integrated circuits not only discourages the use of the design in a portable environment, but also causes overheating, which can lead to soft errors or permanent damage. Hence there is a need to minimize power dissipation which requires accurate estimation of the dissipated power during the design phase.

The main conceptual difficulty in power estimation is that the power depends on the operating *environment* of the circuit; namely, the input vectors being fed in. It is possible to overcome the pattern-dependency problem by using probabilities to describe the set of *all possible logic signals*, and then considering the power resulting from the collective influence of all these signals [2, 6]. This formulation achieves a certain degree of *pattern-independence* that allows one to efficiently estimate the power dissipation. However, in order to achieve good accuracy, one must model the correlations among the signals, which can be very expensive. As a result, these techniques usually trade-off accuracy for speed. Other techniques, based on Monte Carlo Simulation [3, 4], can model the correlation between the circuit primary inputs by selecting appropriate input vectors and automatically take care of correlations between internal signals.

The above techniques were designed primarily to analyze combinational circuits. Only a few techniques have been proposed for sequential circuits. A majority of these techniques decouple the combinational portion of the sequential circuit from the flip-flops and analyze the two separately. The first step is to calculate the switching activity statistics at the flip-flop outputs, which can then be used in the second step to study power consumption in the combinational portion.

†This author is presently at Synopsys Inc., Mountain View, CA.

These techniques can be Monte-Carlo simulation based techniques [5] or techniques based on the solution of non-linear equations [7-10] which model the FSM behavior. When the latch statistics are considered as inputs to the combinational circuit, information on correlation between the state signals is ignored leading to inaccuracies.

In this paper, we propose a solution to this problem that maintains complete information on the state line correlations. This is done by *not* decoupling the latches from the combinational block, and simulating the entire circuit in a Monte Carlo loop until the total power (for the latches and the combinational part) has converged.

In general, all Monte Carlo techniques require a mechanism to generate *mutually independent* samples of the circuit-power, which are then analyzed to determine when the desired accuracy (based on user-specified error-tolerance and degree of confidence) is achieved. The algorithm iterates till the desired accuracy is obtained. Mutually independent samples can be generated by simulating a single copy of a combinational circuit over period of time, and collecting statistics over various *non-overlapping* time intervals. In case of sequential circuits, because of feedback, such a straightforward technique would not result in independent samples, and a more sophisticated approach is required.

In this paper, we present a mechanism for generating mutually independent samples using multiple copies of the circuit which are simulated in parallel, using mutually independent input vector streams. Each copy of the machine results in an independent sample of power, and the samples are then collectively analyzed to determine when to stop the simulation process. Since the circuit is studied as a single entity, the correlations between flip-flop outputs are automatically taken care of.

In most simulation based mechanisms, the numbers of vectors fed into the circuit is fixed *a-priori*. The power measured using this set of vectors is assumed to be the *average* power dissipated, without regard to the variation in power that may be expected when a different set of input vectors is used. There is no guarantee that the vector set used was long enough to be representative of the typical operating environment of the circuit. Our approach offers a solution to this problem by continuing the simulation until the measured standard deviation in power over all the machines is within certain bounds. This gives us the confidence in the power estimate which without having to do exhaustive simulation.

In the following sections, formulate the Monte-Carlo simulation based technique in more detail (section 2), present our approach (section 3), give experimental results (section 4), and conclude with some discussion (section 5).

2. Problem Formulation

In order to capture the average power over long time periods, in this section we will formulate the FSM power estimation problem using probability. It is helpful before we get into this to summarize the final results of this analysis. The two main results are equations (7) and (8). Equation (7) says that if we start the FSM in some initial state X_0, and somehow monitor the mean of the cumulative power over time, then that should converge to the desired power value. It is important to note that this is independent of the initial state. The way to predict the "mean of the cumulative power over time" is to make use of equation (8) which says that we should wait until the variance is small enough, at which time we know that any observed value of P_K is probably close enough to the desired mean.

We start with a comment on the reachability of the state space. In general, the set of useful states of an FSM may not cover the entire state space. If the state space is disconnected, we assume that only one connected subset of the space represents the useful region of operation and contains some *reset state*, to be denoted X_0, which we assume is given. This assumption is quite mild and is generally satisfied in practice.

Since the system is clocked, it is convenient to work with discrete time. In order to take into account the effect of large sets of inputs, one is typically interested in the average power dissipation over long periods of time. Therefore, we will assume that the FSM operates for all time $(-\infty < k < \infty)$.

We consider that the power dissipated by a cell (logic gate or memory element) is due mainly to logic transitions at its output(s). Therefore , the energy drawn for every two consecutive transitions at the cell output i is given by $V_{dd}^2 C_i$. In the simplest cases, the coefficient C_i can be the lumped total capacitance at the cell output (in general, part of C_i can be to V_{dd} and part to ground). However, given a good characterization or extraction methodology, the C_i term can be tuned to represent additional effects such as the short circuit power, interconnect capacitance, internal capacitance charging/discharging, etc.

In any case, if node i makes $n_k(i)$ logic transitions in clock cycle k, then the total energy dissipated in that cycle is given by:

$$e(k) = \frac{1}{2} \sum_{i=1}^{M} V_{dd}^2 C_i n_k(i) \tag{1}$$

where M is the total number of gate/latch output nodes in the circuit. The average power dissipation over K clock cycles is, therefore:

$$P_K = \frac{1}{KT_c} \sum_{k=1}^{K} e(k) \tag{2}$$

In order to study the average power over all time $(-\infty, \infty)$, it is useful to consider a random model of logic signals. We will use **bold font** to represent random quantities. We denote the probability of an event A by $\mathcal{P}\{A\}$ and, if $\mathbf{x}$ is a random variable, we denote its mean by $E[\mathbf{x}]$. An infinite logic signal $x(k)$ can be viewed as a sample of a stochastic process $\mathbf{x}(k)$, consisting of an infinite set of shifted copies of the logic signal. This process, called a *companion process* [5], embodies all the details of the logic signal, including its switching activity. Furthermore, it is *stationary*, i.e., its statistics such as, say, its probability and variance at a given time, do not change with time.

If we (conceptually) construct the companion processes corresponding to all the FSM signals (input, state, output, and internal signals), then we can view the FSM as a system operating on stochastic inputs, consisting of the companion processes $\mathbf{u_1}(k), \mathbf{u_2}(k), \ldots, \mathbf{u_m}(k)$, and having a stochastic state consisting of the processes $\mathbf{x_1}(k), \mathbf{x_2}(k), \ldots, \mathbf{x_n}(k)$. A key fact, due to stationarity of the companion processes, is that the FSM, viewed as a stochastic system as described, is also stationary. Thus any statistics related to the FSM do not change with time.

The energy dissipated per cycle $e(k)$ also leads to a corresponding companion process $\mathbf{e}(k)$ (representing the random energy dissipated in an arbitrarily chosen cycle). This process is also stationary, so that its mean $E[\mathbf{e}(k)]$ is the same for any k, and we denote it by $E[\mathbf{e}]$. We can now write the average power over K cycles as the random variable:

$$\mathbf{P_K} = \frac{1}{KT_c} \sum_{k=1}^{K} \mathbf{e}(k) \tag{3}$$

Since $\lim_{K\to\infty} (1/K) \sum_{k=1}^{K} \mathbf{e}(k) = E[\mathbf{e}]$, then the average power over all time becomes a fixed (non-random) value P, given by:

$$\lim_{K\to\infty} \mathbf{P_K} = P = \frac{E[\mathbf{e}]}{T_c} \tag{4}$$

This result, that the power is equal to the mean energy per cycle divided by cycle length, is of course self evident and well known. The purpose of the above companion process development is to formalize the dependence of this mean on the underlying statistics of the FSM inputs. Notice also that, using (3) and (4), we can write:

$$P = E[\mathbf{P_K}] \tag{5}$$

for any value of $K \geq 1$. If K is viewed as a variable, then $\mathbf{P_K}$ becomes a stochastic process which, again, is stationary, so that $E[\mathbf{P_K}]$ does not depend on K. With this, the FSM power estimation problem may be stated as follows: given statistics of the input vector $\mathbf{U}(k) = [\mathbf{u_1}(k)\ \mathbf{u_2}(k)\ \ldots\ \mathbf{u_m}(k)]$, compute the average power over all time, P.

It is clear from (5) that one possible way of estimating the power is to obtain enough samples of $\mathbf{P_K}$ in order to estimate its mean using Monte Carlo techniques. The problem with this approach is that the samples must be from the *stationary* $\mathbf{P_K}$ process. This requires one to know the statistics $\mathbf{P_K}$ up-front, which are of course unknown - indeed, the mean of $\mathbf{P_K}$ is the average power P which we are looking for. In the next section, we will propose a solution that overcomes this problem, which requires the following mild assumption related to the operation of the FSM:

Assumption 1. *The energy dissipated by the FSM in* *cycle k, $\mathbf{e}(k)$, becomes independent of its initial state at time 0 as $k \to \infty$.*

This assumption is mild because it is generally true in practice that, for all non-periodic logic signals, two values of the signal that are separated by a large number of clock cycles become increasingly unrelated. One necessary condition of this assumption is that the FSM be *aperiodic*. Aperiodicity is implicitly assumed by most previous work on sequential circuits (Markov assumption).

As a result of this assumption, and assuming we start in a certain state X_0 at time 0, we can write:

$$\lim_{k\to\infty} E[\mathbf{e}(k) \mid X_0] = E[\mathbf{e}] \tag{6}$$

so that the process $\mathbf{e}(k)$, which is initially *not* at its stationary state because we have arbitrarily placed the machine in a specific state X_0, eventually becomes stationary, after a long enough time period. From this, and using (3), it follows that:

$$\lim_{K\to\infty} E[\mathbf{P_K} \mid X_0] = E[\mathbf{P_K}] = P \tag{7}$$

from which, and since P is a deterministic value, it is also clear that:

$$\lim_{K\to\infty} \mathrm{Var}[\mathbf{P_K} \mid X_0] = 0 \tag{8}$$

where Var denotes the variance.

These two equations (7) and (8) are the key to our proposed technique, for which the implementation using statistical sampling is given in the next section.

4. Statistical Sampling

Suppose we perform N separate simulation runs of the FSM, all starting from the same initial state X_0, and we drive the different runs by input vector streams that are *independently* chosen. For each simulation run $j = 1, 2, \ldots, N$ and for each clock cycle $k = 1, 2, \ldots$, we can compute the energy consumed per cycle $e^{(j)}(k)$, and the cumulative power up to time K:

$$P_K^{(j)} = \frac{1}{KT_c} \sum_{k=1}^{K} e^{(j)}(k) \tag{9}$$

For a given K, the data set $\left(P_K^{(1)}, P_K^{(2)}, \ldots P_K^{(N)}\right)$ constitutes a *random sample* [12] from the random variable (RV) $\mathbf{P_K}$, conditional on the initial state X_0. For this claim to be true, it is critical that the FSMs are driven with independent input streams, as we have proposed, so that the values $P_K^{(j)}$ are samples of independent RVs. If we define:

$$\mu_N(K) = \frac{P_K^{(1)} + P_K^{(2)} + \cdots + P_K^{(N)}}{N} \tag{10}$$

then it should be clear that, for large N, we have:

$$\mu_N(K) \approx E[\mathbf{P_K} \mid X_0] \tag{11}$$

In our case, we have fixed N because that would be too inefficient to increase the number of simulation runs on the fly. It would seem, therefore, that we do not have a good way of improving the accuracy of (11). This is where (8) comes in. Since the variance of $\mathbf{P_K}$ decreases with K, then the accuracy of (11) improves with K. Thus, if we keep simulating and wait long enough, $\mu_N(K)$ becomes a good estimator of $E[\mathbf{P_K} \mid X_0]$, due to (8), and then if we simulate further and wait longer, $\mu_N(K)$ becomes a good estimator of the desired average power P, due to (7).

The above description gives the essence of our technique. In practice, we use two initial states X_0 and X_1 (see section 4.2), and perform two sets of N simulations each. This is done so that it is easier to check when the two resulting $\mu_N(K)$ waveforms have converged by checking if their difference is close to zero. In the remainder of this section, we give the mechanics of how exactly we determine when the user specified accuracy has been converged (the stopping criterion) and discuss the nature of the input vectors.

4.1 Stopping criterion

Recall the definition of $\mu_N(K)$, given in (10), and let $\sigma_N(K)$ denote the *standard deviation* of the data set $\left(P_K^{(1)}, P_K^{(2)}, \ldots P_K^{(N)}\right)$. According to the *Central Limit Theorem* [12], $\mu_N(K)$ is a sample of a random variable, called the *sample mean*, whose mean is $\mu(K) = E[\mathbf{P_K} \mid X_0]$ and whose distribution approaches the *normal distribution* for large N. The minimum number of samples, N, to satisfy near-normality is typically 30. It is also known that for values of N larger than this one may use $\sigma_N(K)/\sqrt{N}$ as an estimate of the true variance $\sigma(K)$ of the sample mean.

In our implementation, we use $N = 50$, so that the distribution of the sample mean is near-normal, and we can make inferences about the quality of an *individual* sample. We would like to stress that we are not assuming that the power in an individual circuit is distributed near-normally. This work is based on the fact that the distribution of the sample-mean is near-normal.

With $(1-\alpha)$ confidence, it then follows that [13]:

$$-z_{\alpha/2}\sigma(K) \le E[\mathbf{P_K} \mid X_0] - \mu_N(K) \le z_{\alpha/2}\sigma(K) \tag{12}$$

where $z_{\alpha/2}$ is defined so that the area to its right under the standard normal distribution curve is equal to $\alpha/2$. Equation (12) may be rearranged to better accommodate mean estimation, by using:

$$\sigma(K) \approx \frac{\sigma_N(K)}{\sqrt{N}} \tag{13}$$

The above equation is justified for values of N which normalize the sample mean distribution, typically $N \ge 30$, as we have taken. Thus using equations (12) and (13), with confidence $(1-\alpha)$, we have

$$\frac{|E[\mathbf{P_K} \mid X_0] - \mu_N(K)|}{\mu_N(K)} \le \frac{z_{\alpha/2}\sigma_N(K)}{\mu_N(K)\sqrt{N}} \tag{14}$$

If ϵ_1 is a small positive number, and if K is large enough to achieve:

$$\sigma_N(K) \le \frac{\epsilon_1 \mu_N(K)\sqrt{N}}{z_{\alpha/2}} \tag{15}$$

then ϵ_1 places an upper bound on the percentage error of the sample with $(1-\alpha)$ confidence:

$$\frac{|E[\mathbf{P_K} \mid X_0] - \mu_N(K)|}{\mu_N(K)} \leq \frac{z_{\alpha/2}\sigma_N(K)}{\mu_N(K)\sqrt{N}} \leq \epsilon_1 \qquad (16)$$

This may also be expressed as the percent deviation from the population mean $E[\mathbf{P_K} \mid X_0]$:

$$\frac{|E[\mathbf{P_K} \mid X_0] - \mu_N(K)|}{\mu_N(K)} \leq \epsilon_1$$

which translates to:

$$\frac{|\mu_n - E[\mathbf{P_K} \mid X_0]|}{E[\mathbf{P_K} \mid X_0]} \leq \frac{\epsilon_1}{1-\epsilon_1} = \epsilon \qquad (17)$$

Here, ϵ is defined as the user-specified error tolerance, and α (or $1-\alpha$) is the user-specified confidence. Thus equation (15) provides a stopping criterion to yield the accuracy specified in (16) with confidence $(1-\alpha)$.

When K is large enough so that (15) is satisfied, then $\mu_N(K)$ is close enough to the $E[\mathbf{P_K} \mid X_0]$ curve, and we can start to check if it has converged to its steady state value which is the desired average power, P. As mentioned previously, we perform two sets of N simulation runs each, with one set of machines, S_0, starting from an initial state X_0 and the other set S_1, starting from another initial state X_1. This leads to two waveforms $\mu_N^{(0)}(K)$ and $\mu_N^{(1)}(K)$, both of which should eventually converge to P. To better monitor their convergence, we measure the difference $\delta(K) = |\mu_N^{(0)}(K) - \mu_N^{(1)}(K)|$ and the average $\eta(K) = 0.5(\mu_N^{(0)}(K) + \mu_N^{(1)}(K))$. We declare convergence when the relative difference is less than 2ϵ:

$$\delta(K)/\eta(K) \leq 2\epsilon \qquad (18)$$

The value of the average $\eta(K)$ is reported as the power value.

4.2 Input vectors and Initial States

In view of our basic assumption, one requirement on the applied input sequence $U(k)$ is that it not be periodic. Another condition, required for the Monte Carlo estimation to hold, is that the different $U^{(j)}(k)$ sequences used in different simulation runs $j = 1, \ldots, N$ be generated or chosen *independently*. Otherwise, no limitations are placed on the input sequence. The exact way in which the inputs are generated depends on the design and on what information is available about the inputs. The vectors for the results presented in this paper were generated randomly under the assumption that all input patterns are equiprobable.

To carry out the simulations described above, we require two initial states X_0 and X_1 for the FSM. This information may be supplied by the designer, the only limitation being that the two states be in the same connected sub-space. In case no information is available, warm-up simulations may be used to generate a second state which is reachable from the reset state (default $X_0 = [0\ 0\ \ldots\ 0]$).

Table 1.
Results for ISCAS89 circuits. $\alpha = 0.05$ & $\epsilon = 0.05$.

Circuit	Step1 Cycles	Total Cycles	Power mW/MHz	cpu (sec)	% Error
s208	30	40	0.00698	4.92	0.87
s298	30	30	0.00912	3.87	1.33
s344	30	30	0.01846	7.73	0.16
s349	30	30	0.01856	7.80	0.11
s382	30	30	0.01048	4.66	2.14
s386	30	30	0.01620	5.27	0.81
s400	30	30	0.01065	3.59	1.24
s420	30	30	0.00903	4.05	4.27
s444	30	30	0.01172	3.79	2.27
s641	30	30	0.03629	18.16	0.61
s713	30	30	0.03743	17.19	1.21
s820	30	30	0.02831	13.81	3.47
s838.1	30	30	0.01292	9.39	1.65
s953	30	40	0.02458	16.46	2.42
s1238	30	30	0.06347	26.29	-
s1423	30	30	0.07181	29.79	-
s1488	30	80	0.05648	23.93	-
s1494	30	30	0.06018	24.02	-
s5378	30	30	0.23357	72.16	-
s9234.1	30	30	0.28004	134.14	-
s13207.1	30	30	0.35404	179.92	-
s15850.1	30	30	0.51991	243.66	-
s38417	30	30	1.14518	536.70	-
s38584.1	30	40	1.87987	1199.70	-

5. Experimental Results and Analysis

The technique proposed above has been implemented and tested on a number of benchmark circuits. We use an event-driven, gate-level simulator with a scalable delay timing model (based on output capacitance and fanout). During the initial part of the simulation process, we introduce a warm-up period of 30 cycles during which the convergence criterion (equation 15) is not applied. This is to protect from the possibility of observing artificially small variance in the initial few cycles of the simulation. Figure 1 shows a typical plot of the observed variance over the two sets of machines. The variance curve has a steep rise with a gradual fall. Similar plots have been observed for all the circuits considered with all circuits reaching the peak variance in less than 15 cycles. Thus 30 cycles are more than enough to ensure that variance has reached its peak and has started to fall.

We present results for two different set of simulations on the ISCAS89 sequential benchmarks circuits. The first set (Table 1) was run with $\epsilon = 0.05$ and $\alpha = 0.05$, the second set (Table 2) was run with$\epsilon = 0.01$ and $\alpha = 0.01$. We also carried out a long simulation using more than 2.5 million vectors on some of the smaller ISCAS89 benchmarks, to act as a benchmark to compare the accuracy of our results.. The simulation setup was the same as the experiment, except that a warm-up period of 25,000 vectors per machine was used.

In these tables, the column *Step 1 cycles* gives the number of cycles required for equation (15) to be satisfied by both the sets of machines S_0 and S_1. The column *Total Cycles* gives the number of cycles required for equation (18) to be satisfied. As expected the second run with stricter convergence bound takes longer number of cycles to converge. The number of cycles required for the variance of individual samples to converge does not increase much, though the number of cycles required for the two means to converge (equation (18)) is increased. The run-time are on a SUN Sparc5 with 32 MB of memory. The simulation with looser bounds takes much less time, specially for the larger circuits.

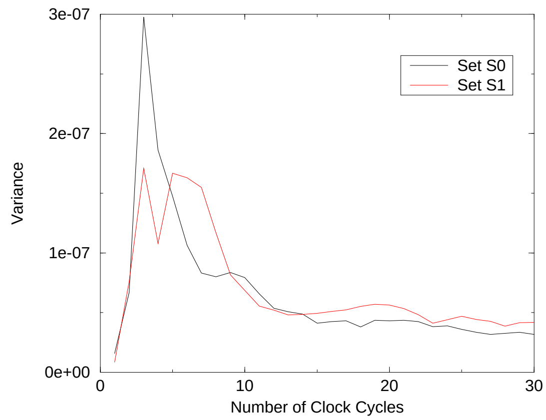

Figure 1. Variance in Power vs Time.

For most of the circuits, the simulation with the tighter convergence bounds, had less error. The absolute percentage error is very low and except for two circuits is less than 3%.

6. Conclusion

We have proposed a simulation-based Monte Carlo method for estimating the power dissipation of sequential circuits which considers correlations in time and space between the inputs, the internal nodes, state nodes, etc. For this reason, this work constitutes a definite improvement over existing methods which have limited ability to handle correlations. The simulation can be driven by user-supplied inputs. Our contribution consists of specifying exactly how the simulation results should be analyzed, to tell when the power measured from the simulation has converged to within user-specified accuracy and confidence. As a result reliable results can be obtained without doing long exhaustive simulations, and hence saving time.

References

[1] R. W. Brodersen, A. Chandrakasan, S. Sheng, "Technologies for personal communications," *1991 Symp. on VLSI circuits*, Tokyo, Japan, pp. 5-9, 1991.

[2] F. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 2, pp. 3¡10-323, February 1993.

[3] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," *31st ACM/IEEE Design Automation Conference*, San Diego, CA, pp. 728–733, 1994.

[4] R. Burch, F. Najm, P. Yang, and T. Trick, "A Monte Carlo approach for power estimation," *IEEE Transactions on VLSI Systems*, vol. 1, no. 1, pp. 63–71, March 1993.

[5] F. Najm, S. Goel, and I. Hajj, "Power Estimation in Sequential Circuits," *32nd ACM/IEEE Design Automation Conference*, San Francisco, CA, pp. 635–640, 1995.

[6] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 446–455, Dec. 1994.

[7] A. A. Ismaeel and M. A. Breuer, "The probability of error detection in sequential circuits using random test vectors," *Journal of Electronic Testing*, vol. 1, pp. 245–256, January 1991.

[8] G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Probabilistic analysis of large finite state machines," *31st ACM/IEEE Design Automation Conference*, San Diego, CA, pp. 270–275, June 6–10, 1994.

[9] J. Monteiro and S. Devadas, "A methodology for efficient estimation of switching activity in sequential logic circuits," *ACM/IEEE 31st Design Automation Conference*, San Diego, CA, pp. 12–17, June 6–10, 1994.

[10] C-Y Tsui, M. Pedram, and A. M. Despain, "Exact and approximate methods for calculating signal and transition probabilities in FSMs," *ACM/IEEE 31st Design Automation Conference*, San Diego, CA, pp. 18–23, June 6–10, 1994.

[11] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd Edition. New York, NY: McGraw-Hill Book Co., 1984.

[12] M. H. DeGroot, *Probability and Statistics*, 2nd Edition. Reading, MA: Addison-Wesley, 1986.

[13] I. R. Miller, J. E. Freund, and R. Johnson, *Probability and Statistics for Engineers*, 4th Edition. Englewood Cliffs, NJ: Prentice-Hall Inc., 1990.

[14] T-L Chou, and K.Roy, " Accurate Power Estimation of CMOS Sequential Circuits," *IEEE Transactions on VLSI Systems*, to appear, Sept. 1996.

Table 2.
Results for ISCAS89 circuits. $\alpha = 0.01$ & $\epsilon = 0.01$.

Circuit	Step1 Cycles	Total Cycles	Power mW/MHz	cpu (sec)	% Error
s208	30	360	0.00690	67.60	0.29
s298	30	460	0.00901	95.62	0.11
s344	30	300	0.01845	107.49	0.11
s349	30	300	0.01856	108.88	0.11
s382	30	450	0.01030	115.57	0.39
s386	30	150	0.01624	37.11	1.06
s400	30	50	0.01061	10.25	0.86
s420	30	60	0.00890	14.41	2.77
s444	30	460	0.01150	97.82	0.35
s641	30	30	0.03562	25.97	1.25
s713	30	50	0.03785	41.25	0.11
s820	30	90	0.02798	57.91	2.27
s838.1	30	60	0.01294	37.18	1.81
s953	30	480	0.02402	316.10	0.08
s1238	30	30	0.06347	35.80	-
s1423	30	90	0.07111	124.45	-
s1488	50	1410	0.05477	1559.64	-
s1494	50	1410	0.05444	1546.28	-
s5378	30	130	0.21475	459.47	-
s9234.1	30	130	0.27458	1060.97	-
s13207.1	30	290	0.35427	3292.10	-
s15850.1	30	220	0.53000	3210.88	-
s38417	30	1070	1.14482	33290.34	-
s38584.1	30	690	1.95901	29236.49	-