McPOWER: A Monte Carlo Approach to Power Estimation

Richard Burch*, Farid Najm*, Ping Yang*, and Timothy Trick†

*Semiconductor Process & Design Center
Texas Instruments Inc., MS 369
Dallas, Texas 75265

†Electrical Engineering Dept.
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Excessive power dissipation in integrated circuits causes overheating and can lead to soft errors and/or permanent damage. The severity of the problem increases in proportion to the level of integration, so that power estimation tools are badly needed for present-day technology. Traditional simulation-based approaches simulate the circuit using test/functional input pattern sets. This is expensive and does not guarantee a meaningful power value. Other recent approaches have used probabilistic techniques in order to cover a large set of inputs patterns. However, they trade-off accuracy for speed in ways that are not always acceptable. In this paper, we investigate an alternative technique that combines the accuracy of simulation-based techniques with the speed of the probabilistic techniques. The resulting method is statistical in nature; it consists of applying randomly-generated input patterns to the circuit and monitoring, with a simulator, the resulting power value. This is continued until a value of power is obtained with a desired accuracy, at a specified confidence level. We present the algorithm and experimental results, and discuss the superiority of this new approach.

1. Introduction

Excessive power dissipation in integrated circuits causes overheating and can lead to soft errors and permanent damage. The severity of the problem increases in proportion to the level of integration. The advent of VLSI has led to much recent work on the estimation of power dissipation *during* the design phase, so that designs can be modified before manufacturing.

Perhaps the most significant obstacle in trying to estimate power dissipation is that the power is *pattern dependent.* In other words, it strongly depends on the input patterns being applied to the circuit. Thus the question "what is the power dissipation of this circuit?" is only meaningful when accompanied with some information on the circuit inputs.

A direct and simple approach of estimating power is to simulate the circuit. Indeed, several *circuit simulation* based techniques have appeared in the literature [1-2]. Given the speed of circuit simulation, these techniques can not afford to simulate large circuits for long-enough input vector sequences to get meaningful power estimates. In order to simplify the problem and improve the speed, the power supply voltage is often assumed to be the same throughout the chip. Thus the power estimation problem is reduced to that of estimating the power supply *currents* that are drawn by the different circuit components. Fast timing or logic simulation can then be used to estimate these currents [3].

We call these approaches *strongly* pattern dependent because they require the user to specify *complete* information about the input patterns. Recently, other approaches have been proposed [4, 5] that only require the user to specify *typical* behavior at the circuit inputs using *probabilities.* These may be called *weakly* pattern dependent. With little computational effort, these techniques allow the user to cover a huge set of possible input patterns. However, in order to achieve good accuracy, one must model the correlations between internal node values, which can be very expensive. As a result, these techniques usually trade-off accuracy for speed. The resulting loss of accuracy is a significant issue that may not always be acceptable to the user.

In this paper, we investigate an alternative approach that combines the accuracy of simulation-based approaches with the weak pattern dependence of probabilistic approaches. The resulting approach is *statistical* in nature; it consists of applying randomly-generated input patterns to the circuit and monitoring, with a simulator, the resulting power value. This is continued until a value of power is obtained with a desired *accuracy*, at a specified *confidence* level. Accuracy can be traded-off to increase speed; however, we will show that high levels of accuracy can be obtained quickly. Since our approach uses a *finite* number of patterns to estimate the power, which really depends on the *infinite* set of possible input patterns, this method belongs to the general class of so-called *Monte Carlo* methods. A most attractive property of Monte Carlo techniques is that they are *dimension independent*, meaning that the *number* of samples required to make a good estimate is independent of the problem size. We will show that this property indeed holds for our approach (see Table 3 in section 4).

Both [4] and [5] use probabilities to compute the power consumed by *individual gates*, which are then summed up to give the total power. In this context, it was observed in [5] that it would be too expensive to estimate the individual gate powers using a simulation with randomly generated inputs. The key to the efficiency of our new approach is that, if one monitors the *total* power directly during the random simulation, sufficient accuracy is obtained in much less time than is required to compute the individual gate powers. The excellent speed performance and the simplicity of the implementation make this a very attractive solution for power estimation.

As a first step, the technique to be presented in this paper will focus on combinational circuits. It turns out that the only issue to be resolved for sequential circuits is the choice of the so-called "setup period" (see section 3.4). Otherwise, there are no fundamental problems in implementing Monte Carlo power estimation for sequential circuits.

An approach similar to this was independently proposed in [6], but the treatment there is not very rigorous and overlooks some important issues. Furthermore, no comparisons were performed with other approaches to show the superiority of the approach. In this paper, we present a rigorous treatment that provides the theoretical justification of this method. We also present experimental results of our implementation and compare it to probabilistic approaches.

2. Overview

In this section, we provide an overall view of our technique, and discuss its superiority to the probabilistic approaches previously proposed [4, 5].

2.1. Monte Carlo power estimation

The block diagram in Fig. 1. gives an overall view of the technique. The *setup* and *sample* blocks are parts of the same logic simulation run, in which the input patterns are *randomly-generated.* The power value at the end of a sampling phase is noted and used to decide whether to *stop* the process or to do another setup-sample run. The decision is made based on the mean and standard deviation of the power values observed at the end of a number of successive iterations.

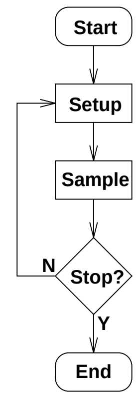

Figure 1. Block diagram overview.

The power is found as the average value of the instantaneous power drawn throughout the sample phase, and *not* during the setup phase. However, the setup phase is a critical component of our approach, and serves two purposes :

(1) In the beginning of the simulation run, the circuit does not switch as often as it typically would at a later time, when switching activity has had time to spread to all the gates. Thus, the circuit is allowed to *get up to speed* during setup. This argument will be made more precise in the section 3, where we also derive an exact value for the setup time.

(2) The values of power observed at the end of successive sample intervals should be samples of independent random variables. This is required in order for the stopping criterion to be correct, and is guaranteed by restarting the random input waveforms at the beginning of the setup phase. The details are given in section 3.

Thus the setup phase guarantees that we are indeed measuring *typical* power, and ensures the *correctness* of the statistical stopping criterion.

2.2. Comparison with probabilistic techniques

There are two distinct advantages of the Monte Carlo approach that make it an excellent choice for power estimation over probabilistic techniques. These are : (1) It achieves *desired* accuracy in reasonable time, avoiding the potential inaccuracy of probabilistic techniques, and (2) the simplicity of the algorithm, making it very easy to implement in existing logic or timing simulation environments.

Probabilistic methods [4, 5] suffer from a speed/accuracy trade-off because they must resolve the correlations between internal circuit nodes. If these correlations *are* taken into account, these methods can be very accurate. This, however, is computationally very expensive and impractical. As a result, fast implementations of these techniques are often inaccurate. It is the aim of this paper to show that the proposed Monte Carlo method is very fast (solving circuits with thousands of gates in a matter of seconds) and also highly accurate (easily within 5% of the total power). Table 3 compares the accuracy of Monte Carlo and probabilistic methods for power estimation.

We also should make the point that the accuracy level in our approach is predictable up-front : the program will work to achieve any level of accuracy desired by the user. Naturally, as higher accuracy is desired, the computational cost starts to increase. However, we will show in section 4 that accuracy levels of 5% are easily and efficiently attainable.

3. Detailed Approach

This section describes the details of the approach. We start out with a rigorous formulation of the problem and show how it reduces to the well-known problem of *mean estimation* in statistics. We then discuss the stopping criterion, and the normality assumption required for it to work. We conclude with a discussion of the setup and sample phases.

3.1. Problem formulation

Consider a digital circuit with m internal nodes (gate outputs). Let $x_i(t)$, $t \in (-\infty, +\infty)$, be the logic signal at node i and $n_{x_i}(T)$ be the number of transitions of x_i in the time interval $(-\frac{T}{2}, +\frac{T}{2}]$. If, in accordance with [7], we consider only the contribution of the charging/discharging current components, the average power dissipated at node i during that interval is $\frac{1}{2}V_{dd}^2 C_i \frac{n_{x_i}(T)}{T}$, where C_i is the total capacitance at i. The total average power dissipated in the circuit during the same interval is :

$$P_T = \frac{V_{dd}^2}{2}\sum_{i=1}^{m} C_i \frac{n_{x_i}(T)}{T}$$

The power rating of a circuit usually refers to its average power dissipation over extended periods of time. We therefore define the *average power dissipation* P of the circuit as :

$$P = \lim_{T\to\infty} P_T = \frac{V_{dd}^2}{2}\sum_{i=1}^{m} C_i \lim_{T\to\infty} \frac{n_{x_i}(T)}{T}$$

The essence of our approach is to estimate P, corresponding to *infinite* T, as the *mean* of several P_T values, each measured over a *finite* time interval of length T. In order to see how this *mean estimation* problem comes about, we must consider a random representation of logic signals as follows.

Corresponding to every logic signal $x_i(t)$, $t \in (-\infty, +\infty)$, we construct a *stochastic process* $\mathbf{x_i}(t)$ as a family of the logic signals $x_i(t+\tau)$, where τ is a random variable. This process has been called the *companion process* of $x_i(t)$ in [8], where the reader may find more details on its construction. For each τ, $x_i(t+\tau)$ is a *shifted* copy of $x_i(t)$. Therefore, observing P_T for $x_i(t+\tau)$ corresponds to measuring the power of $x_i(t)$ over an interval of length T centered at τ, rather than at 0. We can then talk of the *random power* of $\mathbf{x_i}(t)$ over the interval $(-\frac{T}{2}, +\frac{T}{2}]$, to be denoted by :

$$\mathbf{P_T} = \frac{V_{dd}^2}{2}\sum_{i=1}^{m} C_i \frac{\mathbf{n_{x_i}}(T)}{T}$$

where $\mathbf{n_{x_i}}(T)$ is now a random variable. It was shown in [8] that $\mathbf{x_i}(t)$ is *stationary* [12] so that, for *any* T, the expected average number of transitions per second is a constant :

$$E\left[\frac{\mathbf{n_{x_i}}(T)}{T}\right] = \lim_{T\to\infty}\frac{n_{x_i}(T)}{T}$$

where $E[\cdot]$ denotes the *expected value* (mean) operator. As a result, $E[\mathbf{P_T}]$ is the same for any T, and the average power can be expressed as a mean :

$$P = E[\mathbf{P_T}]$$

Thus the power estimation problem has been reduced to that of *mean estimation*, which is a frequently encountered problem in statistics.

In order to apply the above theory, we must ensure that the signals $x_i(t)$ observed throughout the $(\frac{-T}{2}, \frac{+T}{2}]$ interval are *samples* of the *stationary* processes $\mathbf{x_i}(t)$. This requirement will be addressed in section 3.4.

3.2. Stopping criterion

Let us assume that $\mathbf{P_T}$ is *normally distributed* for any T. The theoretical justification and experimental evidence for this assumption will be discussed in the next section. Suppose also that we perform N different simulations of the circuit, each of length T, and form the *sample average* η_T and *sample standard deviation* s_T of the N different P_T values found. Therefore, we have $(1-\alpha)\times 100\%$ *confidence* that $|\eta_T - E[\mathbf{P_T}]| < t_{\alpha/2}s_T/\sqrt{N}$, where $t_{\alpha/2}$ is obtained from the t distribution [9] with $(N-1)$ degrees of freedom. This result can be rewritten as :

$$\frac{|P-\eta_T|}{\eta_T} < \frac{t_{\alpha/2}s_T}{\eta_T\sqrt{N}}$$

Therefore, for a desired *percentage error* ϵ in the power estimate, and for a given confidence level $(1-\alpha)$, we must simulate the circuit until :

$$\frac{t_{\alpha/2}s_T}{\eta_T\sqrt{N}} < \epsilon$$

In order to apply the above theory, we must ensure that the observed P_T values are samples from *independent* $\mathbf{P_T}$ random variables. This requirement will be addressed in section 3.4.

3.3. Normality

A sufficient condition for the normality of $\mathbf{P_T}$ is that (i) m is large and (ii) $\frac{\mathbf{n_{x_i}}(T)}{T}$ are independent. This is true under fairly general conditions irrespective of the individual $\frac{\mathbf{n_{x_i}}(T)}{T}$ distributions (see [10], pp. 188–189).

Another sufficient condition that holds even for small m is as follows. If (i) the consecutive times between identical transitions of $\mathbf{x_i}(t)$ are independent (which, using renewal theory (see [11], pp. 62–63), means that $\frac{\mathbf{n_{x_i}}(T)}{T}$ is normally distributed for large T) and (ii) the $\frac{\mathbf{n_{x_i}}(T)}{T}$ are independent (so that they are also jointly normal (see [12], p. 126) for large T) then $\mathbf{P_T}$ is normal for large T (see [12], p. 144).

To the extent that these conditions are approximately met in practice, the power should be approximately normal. We have found that for a number of benchmark digital circuits [13], the normality assumption is very good, as shown in the *normal scores plots* [9] in Fig. 2. The plot for each circuit corresponds to 1000 evaluations of the average power over a 2.5 μsec interval. Each evaluation covered an average of 50 transitions per primary input.

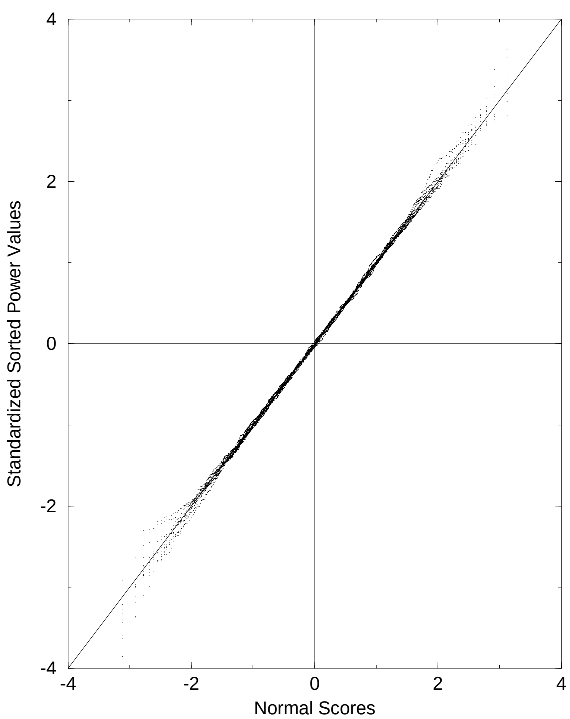

Figure 2. Normal scores plot for the ISCAS-85 circuits.

3.4. Setup and sample

This section deals with the mechanics of how the input patterns are to be generated, when to start and stop measuring a P_T value, and how different P_T values should be obtained. We start by observing that, by stationarity of $\mathbf{x_i}(t)$, the (finite) intervals of width T, over which the P_T values will be measured, need not be centered at the origin. A P_T value may be obtained from any interval of width T, henceforth called a *sampling interval.* However, the following two requirements must be met :

(i) Throughout a sampling interval, the signals $x_i(t)$ must be samples of the *stationary* processes $\mathbf{x_i}(t)$.

(ii) The different P_T samples must be samples from *independent* $\mathbf{P_T}$ random variables.

We will now describe a simulation process that guarantees both of these requirements. Suppose that the circuit primary inputs are at 0 from $-\infty$ to time 0, and then become samples of the stationary processes $\mathbf{x_i}(t)$ in positive time. Consider a primary input driving an inverter with delay t_d. Since its input is a stationary process for $t \geq 0$, its output must be stationary for $t \geq t_d$. By using a simplified timing model for every gate as in [5], we can repeat this argument enough times to obtain the following conclusion: If the maximum delay from the primary inputs to node i is $T_{max,i}$, then the process $\mathbf{x_i}(t)$ becomes stationary for $t \geq T_{max,i}$.

If the maximum delay in the circuit is $T_{max} = \max_i(T_{max,i})$, then the sampling interval may start only after $t \geq T_{max}$. This guarantees that requirement (i) is met. From that time onwards, all internal processes are stationary, and the circuit is in (probabilistic) steady state. We will call the time interval from 0 to T_{max} the *setup* phase. Intuitively, the circuit needs to *get up to speed* before a reliable sample of power may be taken and, as we have shown, the time needed to do that is T_{max}.

If the processes $\mathbf{x_i}(t)$ for $t \geq t_0$ are independent of all occurrences for $t < t_0$, then the state of the circuit for $t \geq t_0 + T_{max}$ is also independent of its state before t_0. This observation is true because the circuit is combinational and T_{max} is the maximum delay. This can be used to guarantee requirement (ii), as follows. Suppose we start the simulation at 0, and measure P_T for the interval $(T_{max}, T_{max} + T]$, and let $t_0 = T_{max} + T$. If we *restart* the primary input processes at t_0 so that they are independent of their past before t_0, then we may measure another value of P_T over the interval $(t_0 + T_{max}, t_0 + T_{max} + T]$. The two resulting values of P_T are samples of *independent* $\mathbf{P_T}$ random variables. As a result, the time axis is divided into successive regions of setup and sampling, as shown in

Fig. 3, where the input processes are restarted at the beginning of every setup phase.

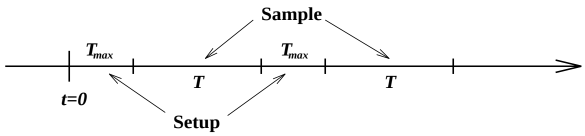

Figure 3. Successive setup and sample phases.

The only remaining task is to describe how the inputs are to be generated. This has to be done in such a way that the input processes, after the start of every setup phase, are independent of the past. This can be done as follows, for every input signal x_i. At the beginning of a setup phase, we use a random number generator to select a logic value for x_i, with appropriate probability $P(x_i)$. We then use another random number generator to decide how long x_i stays in that state before switching. This must assume some distribution for the duration of stay in that state. Once x_i has switched, we use another random number generator to decide how long it will stay in the other state, again using some distribution. Let $F^1_{x_i}(t)$ be the distribution of times spent in the 1 state, and $F^0_{x_i}(t)$ be that of the 0 state. Since computer implementations of random number generators produce sequences of independent random variables, independence between the successive sampling phases is guaranteed.

The probability $P(x_i)$ and distributions $F^1_{x_i}(t)$ and $F^0_{x_i}(t)$ should be supplied by the user. In fact, these parameters represent the way in which the approach is weakly pattern dependent. They also provide the mechanism by which the user can specify any information about typical behavior at the circuit inputs. In order to simplify the user interface, our current implementation does not require the user to actually specify distributions. Rather, we require only two parameters : the average time that an input is high, denoted by $\mu^1_{x_i}$, and the average time that it is low, denoted by $\mu^0_{x_i}$. Based on this, it can be shown [8] that $P(x_i) = \mu^1_{x_i}/(\mu^1_{x_i} + \mu^0_{x_i})$. As for the distributions, and since the choice is immaterial in the absence of additional user information, we choose for simplicity to use *exponential* distributions, so that $F^1_{x_i}(t) = 1 - e^{-t/\mu^1_{x_i}}$ and $F^0_{x_i}(t) = 1 - e^{-t/\mu^0_{x_i}}$. This choice can be easily modified by the user.

4. Experimental Results

The Monte Carlo methods presented in this paper were implemented based on a simple variable delay logic simulator. This program will be referred to as McPOWER. The test circuits to be used in this section are the benchmarks presented at ISCAS in 1985 [13]. These circuits are combinational logic circuits and Table 1 presents the number of inputs, outputs, and gates in each.

Table 1. The ISCAS-85 benchmark circuits.

Circuit	#inputs	#outputs	#gates
c432	36	7	160
c499	41	32	202
c880	60	26	383
c1355	41	32	546
c1908	33	25	880
c2670	233	140	1193
c3540	50	22	1669
c5315	178	123	2307
c6288	32	32	2406
c7552	207	108	3512

We will compare the performance of McPOWER to that of probabilistic methods and substantiate the claims of section 2.2 that McPOWER has better accuracy and competitive simulation times. DENSIM [5, 8] is an efficient probabilistic simulation program that gives the *average switching frequency* (called *transition density* in [5, 8]) at every circuit node. These density values can be used to give an estimate of the total power dissipation.

Table 2 compares the performance of DENSIM, when used to estimate total power, to that of McPOWER. In both programs, every primary input had a signal probability of 0.5 and a transition density of 2e7 transitions per second (corresponding to an average frequency of 10MHz). For McPOWER, a maximum error of 5% with 99% confidence was specified. As mentioned in section 3, McPOWER performs one long simulation that is broken into setup and sampling regions. The delays of the circuit determine the length of each setup region; however, the length of a sampling region is specified by the user. For Table 2, the sampling region was set to 2.5 micro-seconds (abbreviated μs), which allows an average of 50 transitions per sampling interval on each input. The column labeled LOGSIM gives our best estimates of the power dissipation of these circuits. LOGSIM is a logic simulator using models identical to those in DENSIM and McPOWER. The values in the LOGSIM column were obtained from extremely long runs of LOGSIM using randomly generated inputs consistent with the input descriptions used by DENSIM and McPOWER.

As seen from the table, McPOWER is consistently and highly accurate, while DENSIM has significant errors for some circuits. Although DENSIM is frequently faster, McPOWER's reliable accuracy makes it a more attractive approach for power estimation.

Table 2. Power and Time results for the ISCAS-85 circuits. Time is in cpu seconds on a SUN SparcStation1. McPOWER is based on 5% error, 99% confidence, & 2.5μs sampling region.

Circuit	Power			Cpu Time	
Name	DENSIM	LOGSIM	McPOWER	DENSIM	McPower
c432	0.974 mW	1.165 mW	1.17 mW	0.7 sec	2.5 sec (3.6X)
c499	1.977 mW	2.048 mW	2.13 mW	0.8 sec	2.2 sec (2.8X)
c880	2.086 mW	2.829 mW	2.81 mW	1.4 sec	3.8 sec (2.7X)
c1355	3.695 mW	5.735 mW	5.65 mW	1.9 sec	3.6 sec (1.9X)
c1908	5.154 mW	9.734 mW	9.77 mW	3.1 sec	5.6 sec (1.8X)
c2670	7.319 mW	11.438 mW	11.24 mW	4.5 sec	7.1 sec (1.6X)
c3540	9.235 mW	15.328 mW	15.25 mW	5.8 sec	12.2 sec (2.1X)
c5315	15.471 mW	24.102 mW	23.66 mW	8.5 sec	21.9 sec (2.6X)
c6288	31.941 mW	78.883 mW	75.53 mW	7.5 sec	40.4 sec (5.4X)
c7552	23.156 mW	40.006 mW	38.78 mW	12.4 sec	24.7 sec (2.0X)

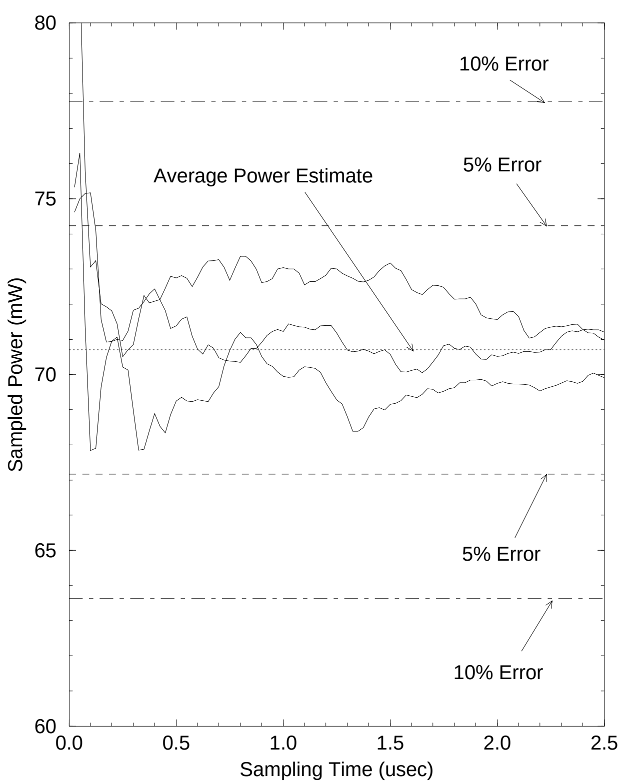

Figure 4. McPOWER convergence results for c6288.

Typical convergent behavior of McPOWER is shown in Fig. 4. The figure shows the power from three different iterations converging to the average power for c6288, one of the most complex ISCAS circuits.

Care must be taken in drawing conclusions from a single run of McPOWER. Since it uses random input vectors, the speed of convergence and the error in the power estimate depend on the initialization of the random number generator. This is illustrated in Table 3, which shows the statistics obtained from one thousand McPOWER runs. The minimum, maximum, and average number of iterations required per run for 5% accuracy with 99% confidence are given. Notice that the average number of iterations required to converge does *not* increase with the circuit size. This confirms the *dimension independence* property of this approach which, as pointed out in the introduction, is a common feature of Monte Carlo methods. Also shown in the table are the percentage number of runs for which the error was greater than 5%, which, as expected, is less than 1% in all cases.

Even smaller execution times are possible if the desired accuracy and confidence levels are relaxed. Table 4 compares DENSIM to a single run of McPOWER with 95% confidence, 20% accuracy, and a sampling region of 625 nano-seconds. As in Table 1, each input has a signal probability of 0.5 and a transition density of 2e7 transitions per second. With these parameters, McPOWER shows competitive speed and still exhibits superior accuracy.

Table 3. Statistics from 1000 McPOWER runs.

Name	Min	Max	Avg	%>5% Err	Avg Cpu Time
c432	3	15	8.0	0.9%	3.2 sec
c499	3	7	3.9	0.0%	2.9 sec
c880	3	14	6.7	0.4%	3.9 sec
c1355	3	7	4.0	0.0%	4.8 sec
c1908	3	11	5.6	0.3%	10.3 sec
c2670	3	10	5.2	0.1%	12.4 sec
c3540	3	13	6.0	0.0%	18.3 sec
c5315	3	7	4.1	0.0%	22.4 sec
c6288	3	6	3.8	0.0%	50.4 sec
c7552	3	10	5.5	0.0%	44.4 sec

5. Conclusions

We have presented a Monte Carlo based power estimation method. Randomly generated input waveforms are applied to the circuit using a logic/timing simulator and the cumulative value of total power is monitored. The simulation is stopped when sufficient accuracy is obtained with specified confidence. The statistical stopping criterion was discussed, along with experimental results from our prototype implementation McPOWER.

We have shown that Monte Carlo methods are, in general, better than probabilistic methods for the estimation of power since they achieve superior accuracy with comparable speeds. They are also easier to implement and can be added to existing timing or logic simulation tools. Furthermore, the accuracy can be specified up-front with any desired confidence.

Feedback circuits present a severe problem for probabilistic methods. Monte Carlo methods are based on simple timing or logic simulation techniques and, therefore, experience very few difficulties with feedback circuits. The only unresolved problem is to determine the length of the setup region, but we feel that good heuristics can be developed for this. Future research will focus on developing such heuristics, thus generalizing Monte Carlo techniques to handle any logic circuit.

Although we have clearly demonstrated the superiority of Monte Carlo methods for power estimation, it is not clear that they will be better than probabilistic methods for other applications, such as estimating the power supply current waveforms. Future research will be aimed at exploring this and other applications of the Monte Carlo approach.

References

[1] S. M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 5, pp. 889–891, Oct. 1986.

[2] G. Y. Yacoub and W. H. Ku, "An accurate simulation technique for short-circuit power dissipation based on current component isolation," *IEEE International Symposium on Circuits and Systems*, pp. 1157–1161, 1989.

Table 4. Power and Time results for the ISCAS-85 circuits. Time is in cpu seconds on a SUN SparcStation1. McPOWER is based on 20% error, 95% confidence, & 625ns sampling region.

Circuit	Power			Cpu Time	
Name	DENSIM	LOGSIM	McPOWER	DENSIM	McPower
c432	0.974 mW	1.165 mW	1.10 mW	0.7 sec	0.7 sec (1.0X)
c499	1.977 mW	2.048 mW	2.04 mW	0.8 sec	0.9 sec (1.1X)
c880	2.086 mW	2.829 mW	2.91 mW	1.4 sec	1.0 sec (0.7X)
c1355	3.695 mW	5.735 mW	5.41 mW	1.9 sec	1.5 sec (0.8X)
c1908	5.154 mW	9.734 mW	9.27 mW	3.1 sec	2.2 sec (0.7X)
c2670	7.319 mW	11.438 mW	10.83 mW	4.5 sec	2.7 sec (0.6X)
c3540	9.235 mW	15.328 mW	14.28 mW	5.8 sec	4.8 sec (0.8X)
c5315	15.471 mW	24.102 mW	22.65 mW	8.5 sec	6.4 sec (0.8X)
c6288	31.941 mW	78.883 mW	71.48 mW	7.5 sec	13.2 sec (1.8X)
c7552	23.156 mW	40.006 mW	37.88 mW	12.4 sec	9.2 sec (0.7X)

[3] A-C. Deng, Y-C. Shiau, and K-H. Loh, "Time domain current waveform simulation of CMOS circuits," *IEEE International Conference on Computer-Aided Design*, pp. 208–211, Nov. 7–10, 1988.

[4] M. A. Cirit, "Estimating dynamic power consumption of CMOS circuits," *IEEE International Conference on Computer-Aided Design*, pp. 534–537, Nov. 9–12, 1987.

[5] F. Najm, "Transition density, a stochastic measure of activity in digital circuits," *28th ACM-IEEE Design Automation Conference*, San Francisco, CA, pp. 644–649, June 17–21, 1991.

[6] C. M. Huizer, "Power dissipation analysis of CMOS VLSI cir-cuits by means of switch-level simulation," *IEEE European Solid State Circuits Conference*, pp. 61–64, Grenoble, France, 1990.

[7] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, pp. 468–473, Aug. 1984.

[8] F. Najm, "Transition density, a new measure of activity in digital circuits," *To appear in IEEE Transactions on Computer-Aided Design*, 1992.

[9] I. R. Miller, J. E. Freund, and R. Johnson, *Probability and Statistics for Engineers.* Englewood Cliffs, NJ: Prentice Hall, 1990.

[10] A. Hald, *Statistical Theory with Engineering Applications.* New York: John Wiley & Sons, 1952.

[11] S. M. Ross, *Stochastic Processes.* New York: John Wiley & Sons, 1983.

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

[13] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," *IEEE International Symposium on Circuits and Systems*, pp. 695–698, June 1985.