Estimation of State Line Statistics in Sequential Circuits

VIKRAM SAXENA, FARID N. NAJM, and IBRAHIM N. HAJJ
University of Illinois at Urbana-Champaign

In this article, we present a simulation-based technique for estimation of signal statistics (switching activity and signal probability) at the flip-flop output nodes (state signals) of a general sequential circuit. Apart from providing an estimate of the power consumed by the flip-flops, this information is needed for calculating power in the combinational portion of the circuit. The statistics are computed by collecting samples obtained from fast RTL simulation of the circuit under input sequences that are either randomly generated or independently selected from user-specified pattern sets. An important advantage of this approach is that the desired accuracy can be specified up front by the user; with some approximation, the algorithm iterates until the specified accuracy is achieved. This approach has been implemented and tested on a number of sequential circuits and has been shown to handle very large sequential circuits that can not be handled by other existing methods, while using a reasonable amount of CPU time and memory (the circuit s38584.1, with 1426 flip-flops, can be analyzed in about 10 minutes).

Categories and Subject Descriptors: B.6.3 [**Logic Design**]: Design Aids—*Simulation*; B.7.2 [**Integrated Circuits**]: Design Aids—*Simulation*; B.8 [**Hardware**]: Performance and Reliability—*Performance Analysis and Design Aids*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-Aided Design*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Power estimation, signal probability, transition density, switching activity, sequential circuit, finite-state machine, signal statistics

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 [Brodersen et al. 1991]. Indeed, excessive power dissipation in integrated circuits not only discourages the use of the design in a portable environment, but also causes overheating,

This work was supported in part by Intel Corp., Digital Equipment Corp., and the Semiconductor Research Corp.
Authors' addresses: V. Saxena, AccelChip Inc., Schaumburg, IL; F. N. Najm, University of Toronto, Department of ECE, 10 Kings College Road, Toronto, Ontario, Canada M5S 3G4; email: f.najm@utoronto.ca; I. N. Hajj, Dean of the Faculty of Engineering and Architecture, American University of Beirut, Beirut, Lebanon.

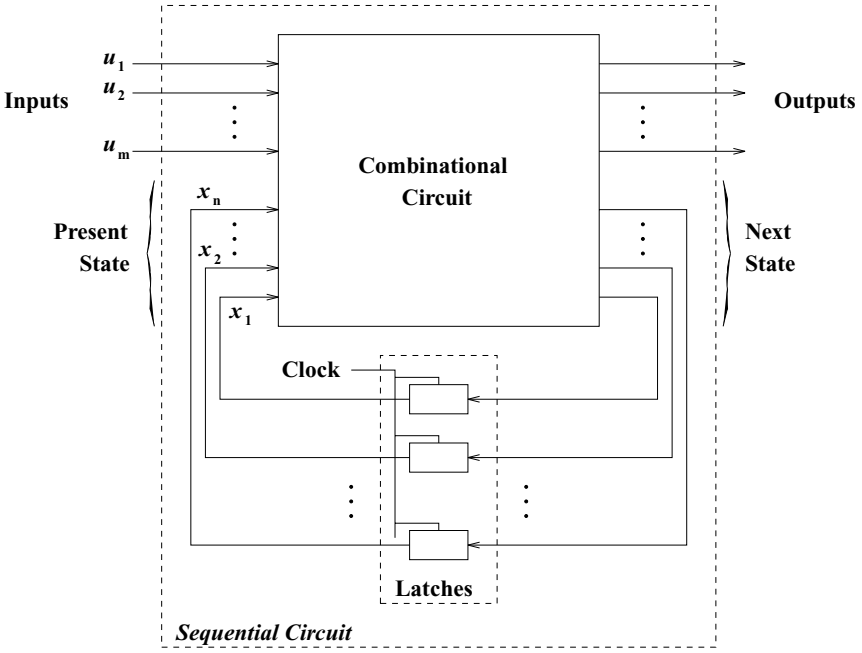

Fig. 1. An FSM model of a sequential logic circuit.

which can lead to soft errors or permanent damage. Hence there is a need to accurately estimate the power dissipation of an IC during the design phase.

The main conceptual difficulty in power estimation is that the power depends on the input signals driving the circuit. Simply put, a more active circuit will consume more power. Thus one straightforward method of power estimation is to simulate the design over all possible inputs, compute the power dissipated under each input, and average the results. However, such an approach is prohibitively expensive. Thus the main difficulty in power estimation is that the power is input pattern-dependent.

It is possible to overcome the pattern-dependency problem by using probabilities to describe the set of all possible logic signals, and then studying the power resulting from the collective influence of all these signals. This formulation achieves a certain degree of pattern-independence that allows one to efficiently estimate the power dissipation. Most recently proposed power estimation tools [Najm 1994] are based on such a probabilistic approach, but are limited to combinational circuits. Only a few techniques have been proposed for sequential circuits, and they are reviewed in the next section.

We consider that the circuit has the popular and well-structured design style of a *synchronous sequential circuit*, as shown in Figure 1. In other words, it consists of flip-flops driven by a common clock and combinational logic blocks whose inputs (outputs) are flip-flop outputs (inputs). Therefore, the average power dissipation of the circuit can be broken down into the power consumed by the flip-flops and that consumed by combinational logic blocks. This provides a convenient way to decouple the problem and simplify the analysis.

In this article, we present a statistical estimation technique for collecting signal statistics (switching activity and signal probability) at the flip-flop outputs. This work extends and improves the preliminary work proposed in Najm et al. [1995]. The statistics are computed by collecting samples obtained from fast register-transfer-level (RTL) simulation of the circuit under input sequences that are either randomly generated or independently selected from user-specified pattern sets. Given these, it is then possible to use any of the existing combinational circuit techniques to compute the power of the combinational circuit. The use of an RTL or zero-delay simulator does not affect the accuracy of the power estimate, since it is assumed the flip-flops are edge-triggered and filter out any glitches or hazards that may exist at their inputs.

In the following sections, we give some background and review of previous approaches (Section 2), formulate the problem in more detail (Section 3), present our approach (Section 4), give experimental results (Section 5), and conclude with some discussion (Section 6). Furthermore, a number of theoretical results are presented and summarized in Appendix A.

2. BACKGROUND

Let $u_1, u_2, \ldots, u_m$ be the primary input nodes of a sequential logic circuit, as shown in Figure 1, and let $x_1, x_2, \ldots, x_n$ be the present state lines. For simplicity of presentation, we have assumed that the circuit contains a single clock that drives a bank of edge-triggered flip-flops. On the falling edge of the clock, the flip-flops transfer the values at their inputs to their outputs. The inputs u_i and the present state values determine the next state values and the circuit outputs, so that the circuit implements a finite state machine (FSM).

Most existing power estimation techniques handle only combinational circuits [Najm 1994] and require information on the circuit input statistics (transition probabilities, etc.). To allow extension to sequential circuits, it is therefore sufficient to compute statistics of the flip-flop outputs (and corresponding flip-flop power). Other existing techniques would then be applied to compute the power consumed in the combinational block.

We briefly survey the few recently proposed techniques for estimating the power in sequential circuits. All proposed techniques that handle sequential circuits [Ismaeel and Breuer 1991; Hachtel et al. 1994; Monteiro and Devadas 1994; Tsui et al. 1994] make the simplifying assumption that the FSM is Markov [Papoulis 1984], so that its future is independent of its past once its present state is specified.

Some of the proposed techniques compute only the probabilities (signal and transition) at the flip-flop outputs, whereas others also compute the power. The approach in Ismaeel and Breuer [1991] solves directly for the transition probabilities on the present state lines using the Chapman–Kolmogorov equations [Papoulis 1984], which is computationally too expensive. Another approach that also attempts a direct solution of the Chapman–Kolmogorov equations is given in Hachtel et al. [1994]. Although it is more efficient, it remains quite expensive, so that the largest test case presented contains less than 30 flip-flops.

Better solutions are offered by Monteiro and Devadas [1994] and Tsui et al. [1994], which are based on solving a nonlinear system that gives the present state line probabilities, as follows. Given probabilities $p_{u_1}, \ldots, p_{u_m}$ at the input lines, let a vector of present state probabilities $P_{p.s.} = [p_{x_1}, \ldots, p_{x_n}]$ be applied to the combinational logic block. Assuming the present state lines are independent, one can compute a corresponding next state probability vector as $F(P_{p.s.})$. The function $F(\cdot)$ is a nonlinear vector-valued function that is determined by the Boolean function implemented by the combinational logic.

In general, if the next state probabilities form a vector $P_{n.s.}$, then $P_{n.s.} \neq F(P_{p.s.})$, because the flip-flop outputs are not necessarily independent. Both methods [Monteiro and Devadas 1994; Tsui et al. 1994] make the independence assumption $P_{n.s.} \approx F(P_{p.s.})$. Finally, since $P_{n.s.} = P_{p.s.}$ due to the feedback, they obtain the state line probability values by solving the system $P = F(P)$. This system is solved using the Newton–Raphson method in Monteiro and Devadas [1994], and using the Picard–Peano iteration method in Tsui et al. [1994].

One problem with this approach is that it is not clear that the system $P = F(P)$ has a unique solution. Being nonlinear, it may have multiple solutions, and in that case it is not clear which is the correct one. Another problem is the independence assumption which need not hold in practice, especially in view of the feedback. Both techniques try to correct for this. In Monteiro and Devadas [1994], this is done by accounting for m-wise correlations between state bits when computing their probabilities. This requires 2^m additional gates and can get very expensive. Nevertheless, they show good experimental results. The approach in Tsui et al. [1994] is to unroll the combinational logic block k times. This is less expensive than Monteiro and Devadas [1994], and the authors observe that with $k = 3$ or so, good results can be obtained. Finally, in order for the FSM to be Markov, its input vectors must be independent and identically distributed, which is another assumption that also may not hold in practice.

A new simulation-based approach was introduced by the authors in Najm et al. [1995] that makes no assumptions about the FSM behavior (Markov or otherwise), makes no independence assumption about the state lines, and allows the user to specify the desired accuracy and confidence to be achieved in the results; with some approximation, the algorithm iterates until the specified accuracy is achieved. The only assumption made (which is given in the next section) has to do with the autocovariance of the logic signals, which is mild and generally true for all but periodic logic signals. The method involves collecting statistics from a number of parallel simulations of the same circuit, each of which is driven by an independent set of vectors. The statistics gathered are used to determine the state line probabilities. In this article, we extend this approach to compute the latch switching activity in addition to the probability, and we provide an improved convergence criterion that significantly improves the run-time with no significant loss of accuracy.

3. PROBLEM FORMULATION

Since the system is clocked, it is convenient to work with discrete time, so that the FSM inputs at time k, $u_i(k)$, and its present state at that time $x_i(k)$,

determine its next state $x_i(k+1)$, and its output. 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 assume that the FSM operates for all time $(-\infty < k < \infty)$. An infinite logic signal $x(k)$ can be characterized by two measures: *signal probability* $P(x)$ is the fraction of clock cycles (time units) in which the signal is high, and *transition density* $D(x)$ is the average number of logic transitions per clock cycle. These measures are formally presented in Appendix A, where it is also shown that $D(x) = P(\mathcal{T}_x)$, where $\mathcal{T}_x(k)$ is another logic signal derived from $x(k)$ so that $\mathcal{T}_x(k) = 1$ only in those cycles where $x(k)$ makes a transition; that is,

$$\mathcal{T}_x(k) = \begin{cases} 1, & \text{if } x(k) \neq x(k-1); \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

It should be stressed that the result $D(x) = P(\mathcal{T}_x)$ is true only for discrete-time logic signals, that is, for signals that make at most one transition per clock cycle, so that they are glitch-free. In this article, we are mainly concerned with the flip-flop outputs which are obviously glitch-free, so that this result is relevant.

In order to study the properties of a logic signal over $(-\infty, \infty)$, it is useful to consider a random model of logic signals. We 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, which we call a *companion process*, embodies all the details of the logic signal, including its probability and density. Details and basic results related to the companion process are given in Appendix A.2 as an extension of previous continuous-time work [Najm 1993b]. Specifically, the companion process is stationary, and for any time instant k, the probability that $\mathbf{x}(k)$ is high is equal to the signal probability of the logic signal:

$$\mathcal{P}\{\mathbf{x}(k) = 1\} = P(x). \tag{2}$$

This result holds for any logic signal. If we (conceptually) construct the companion processes corresponding to the FSM 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 $\mathbf{x_1}(k), \mathbf{x_2}(k), \ldots, \mathbf{x_n}(k)$. Given statistics of the input vector $\mathbf{U}(k) = [\mathbf{u_1}(k)\, \mathbf{u_2}(k) \cdots \mathbf{u_m}(k)]$, one would like to compute some statistics of the state vector $\mathbf{X}(k) = [\mathbf{x_1}(k)\, \mathbf{x_2}(k)\, \cdots\, \mathbf{x_n}(k)]$.

Before going on, we need to make one mild assumption related to the covariance of the process $\mathbf{X}(k)$:

ASSUMPTION 1. *The state of the machine at time 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 nonperiodic logic signals, two values of the signal that are separated by a large number of clock cycles become increasingly uncorrelated. One necessary condition of this assumption is that the FSM be *aperiodic*, that is, that it does not

cycle through a repetitive pattern of states. Aperiodicity is implicitly assumed by most previous work on sequential circuits. Specifically, whenever an FSM is assumed Markov (in which case aperiodicity becomes equivalent to the above assumption) the FSM is usually also assumed to be aperiodic.

Before leaving this section, we consider the question of exactly what statistics of $\mathbf{X}(k)$ are required in order to estimate the power. These statistics must be sufficient to compute the combinational circuit power. Many techniques for combinational circuit power estimation [Najm 1994] require the signal probability and transition density at every input (for discrete-time signals, knowing the transition density is equivalent to knowing the transition probability). Since the power consumed in the flip-flops can also be derived from $D(x_i)$, then the state line $P(x_i)$ and $D(x_i)$ can be sufficient to compute the power for the whole circuit. An algorithm for computing these statistics is presented in the next section.

4. COMPUTING STATE LINE STATISTICS

We propose to obtain the state line statistics by performing Monte Carlo logic simulation of the design using a high-level functional description, say, at the register-transfer-level, and computing the probabilities from the large number of samples produced. High-level simulation can be done very fast, so that one can afford to simulate a large number of cycles. However, we need to define a simulation setup and a mechanism to determine the length of the simulation necessary to obtain meaningful statistics. It is also important to correctly choose the input vectors used to drive the simulation. These issues are discussed below.

4.1 Simulation Setup

We first discuss the estimation of the state line probability $P(x_i)$. Suppose the FSM is known to be in some state X_0 at time 0. Using (2), and given Assumption 1, we have that for any state signal x_i,

$$\lim_{k\to\infty} \mathcal{P}\{\mathbf{x_i}(k) = 1 \mid \mathbf{X}(0) = X_0\} = \lim_{k\to\infty} \mathcal{P}\{\mathbf{x_i}(k) = 1\} = P(x_i).$$

For brevity, we denote the above conditional probability by

$$P_k(x_i \mid X_0) \stackrel{\triangle}{=} \mathcal{P}\{\mathbf{x_i}(k) = 1 \mid \mathbf{X}(0) = X_0\}$$

so that

$$\lim_{k\to\infty} P_k(x_i \mid X_0) = P(x_i). \tag{3}$$

Our method consists of estimating $P_k(x_i \mid X_0)$ for increasing values of k until convergence (according to (3)) is achieved. To accomplish this, we perform a number of simulation runs of the circuit, in parallel, starting from some state X_0, and drive the simulations with input vector streams that are consistent with the statistics of $\mathbf{U}(k)$. Each simulation run is driven by a separate independently chosen input vector stream, and results in a logic waveform $x_i^{(j)}(k)$, $k = 0, 1, 2, \ldots$, where $j = 1, 2, \ldots, N$ designates the run number, and N is the

number of simulation runs. If we average the results at every time k we obtain an estimate of the probability at that time as follows,

$$p_i^{(N)}(k) = \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}(k).$$

From the law of large numbers, it follows that

$$\lim_{N\to\infty} p_i^{(N)}(k) = P_k(x_i \mid X_0).$$

We do not actually have to perform an infinite number of runs. Using established techniques for the estimation of proportions [Miller and Johnson 1990], we can predict how many runs to perform in order to achieve some user-specified error-tolerance (ϵ) and confidence ($1-\alpha$) levels. Specifically, it can be shown [Najm 1993a] that if we want $(1-\alpha)\times 100\%$ confidence that

$$\left|p_i^{(N)}(k) - P_k(x_i \mid X_0)\right| < \epsilon, \tag{4}$$

then we must perform at least $N \geq \max(N_1^2, N_2^2, N_3^2)$ runs, where

$$N_1 = \frac{z_{\alpha/2}}{2\epsilon}, \quad N_2 = \frac{z_{\alpha/2}\sqrt{2\epsilon+0.1} + \sqrt{(\epsilon+0.1)z_{\alpha/2}^2 + 3\epsilon}}{2\epsilon},$$
$$\text{and} \quad N_3 = \frac{\sqrt{63} + z_{\alpha/2}}{2\sqrt{\epsilon}},$$

and where $z_{\alpha/2}$ is a real-valued function of α, defined as follows. Let $\mathbf{z}$ be a random variable with a standard normal distribution, that is, a normal distribution with mean 0 and variance 1. Then, for a given α, $z_{\alpha/2}$ is defined as the real number for which

$$\mathcal{P}\{\mathbf{z} > z_{\alpha/2}\} = \alpha/2.$$

The value of $z_{\alpha/2}$ can be obtained from the **erf**$(\cdot)$ function available on most computer systems. For instance, $z_{\alpha/2} = 1.96$ for 95% confidence (i.e., $\alpha = 0.05$), and $z_{\alpha/2} = 2.575$ for 99% confidence. From the above equations, it can be seen that 490 runs are enough to obtain a result with accuracy $\epsilon = 0.05$ and 95% confidence.

From the user-specified ϵ and α, the required value of N can be found up front. Given this, we initiate N parallel simulations of the FSM and for each state signal x_i obtain waveforms representing $P_k(x_i \mid X_0) \approx p_i^{(N)}(k)$ for increasing k values. The same methodology can be used to estimate $D(x_i)$. During the simulation, statistics for estimation of the state transition density are also collected, along with the statistics for state line probability estimation. This results in another set of waveforms for each state signal x_i representing $D_k(x_i \mid X_0) = P_k(\mathcal{T}_{x_i} = 1 \mid X_0)$, where $\mathcal{T}_{x_i}(k)$ is defined in (1).

The remaining question is how to determine when k is large enough so that $P_k(x_i \mid X_0)$ and $D_k(x_i \mid X_0)$ can be said to have converged to $P(x_i)$ and $D(x_i)$. This is discussed in the next section.

4.2 Convergence in Time

Accurate determination of convergence in time (k) can be computationally very expensive. This is not because the time to converge is long, but because studying the system dynamics that determine convergence is very expensive. For instance, even in the relatively simpler case when the system is assumed Markov, convergence is related to the eigenvalues of the system matrix, whose size is exponential in the number of flip-flops. To overcome this difficulty, we use a heuristic technique to efficiently check for convergence. Simply stated, we monitor the waveform values, over time, until convergence is detected. In order for this simple approach to work well in practice, we make careful choices for the specific ways in which the waveforms are monitored and convergence is checked, as we describe. The resulting method, which we have found works quite well in practice, has several important features: (1) monitor two waveforms instead of one, (2) check both the average and difference of the two waveforms over a time window, and (3) use a low-pass filter to remove the noise in the waveforms. These are explained below, where we restrict the discussion to the probability waveforms since the treatment of the density waveforms is similar.

4.2.1 *Two Waveforms.* Checking convergence of one waveform, say, $P_k(x_i \mid X_0)$, may be done by simply monitoring the waveform values until they have "leveled off" and remained steady for some length of time. By itself, this simple approach is not advisable because it is possible for a waveform to level off for some time and then change again before reaching its steady-state value. In order to reduce the chance of this type of error, we monitor two versions of the P_k waveform for each x_i, and check on convergence by looking at both of them. This is done by considering two different initial states denoted X_0 and X_1, as follows.

It is clear from Equation (3) that the choice of the initial state does not affect the final result. It may affect the rate of convergence, but not the final probability or density values. The only requirement, in order for the error tolerance and confidence results to be valid, is that all the N simulation runs start in the same state. Thus, we perform two sets of simulation runs of the machine, each consisting of N machines running in parallel. Each of the N machines in a set starts in the same initial state, but different initial states, X_0 and X_1, are used for the two sets (the mechanism for determining X_0 and X_1 is presented in Section 4.3). Each machine is driven by an independently selected (see Section 4.5) input stream so that the observed data from the different machines constitute a random sample. For each state line signal, statistics are collected from each of the N parallel simulations in a set. This results in two waveforms for the steady-state probability, $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$, for increasing k values.

Is it possible to further improve the technique by examining three or more waveforms? It may be, but we have observed that the use of two waveforms gives sufficient accuracy. The second waveform basically gives a second opinion, and we have not found a need for a third.

4.2.2 *Time Window.* We use two measures to check on the convergence of the pairs of P_k waveforms. Since both $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ should

converge to $P(x_i)$, we monitor both their *difference* (δ_k) and their *average* (μ_k); where $\delta_k(x_i)$ is defined as $|P_k(x_i \mid X_0) - P_k(x_i \mid X_1)|$ and $\mu_k(x_i)$ is defined as $(P_k(x_i \mid X_0) + P_k(x_i \mid X_1))/2$. When $\delta_k(x_i)$ remains within $\pm\epsilon$ of 0 and $\mu_k(x_i)$ remains within $\pm\epsilon$ of some fixed value, for a certain time window, we consider that $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ have converged to their steady-state $P(x_i)$ value. We have experimented with various time window sizes, and found that a window of just three cycles is sufficient.

During the simulation, we simultaneously obtain another set of two waveforms corresponding to the transition density, $D_k(x_i \mid X_0)$ and $D_k(x_i \mid X_1)$, for each state signal x_i. The convergence criteria used for $P(x_i)$ presented above are also applied to determine the convergence of $D(x_i)$. A state signal x_i is declared converged when both $P(x_i)$ and $D(x_i)$ have converged.

4.2.3 *Filter.* The combination of the two features presented above gives good results in practice, but we have found that it sometimes takes longer than it should to observe convergence. By this we mean, for instance, that both waveforms $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ will be found to "hover" for a long time around the same value. They will have effectively converged, but their constant fluctuations around the steady-state value impede the convergence check. The fluctuations have the character of random noise and in some cases they may simply be due to slow system dynamics. In any case, it is clear that their removal is imperative in order to get a faster convergence check.

To achieve this, the waveforms are *filtered* before the convergence criteria are applied to them. We use a linear phase, ideal low-pass filter with an empirically selected f_cT equal to 0.02, in conjunction with a Hamming window of width 100. The impulse response of the filter is:

$$h[n] = \begin{cases} \dfrac{\sin[2\pi 0.02(n-50)](0.54 - 0.46\cos[2\pi n/100])}{\pi(n-50),} & \text{if } 0 \le n \le 100; \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

The first sidelobe of the Hamming window is 41 dB below the main lobe. As a result the negative component in its frequency response is negligible in comparison with the other filtering windows such as the Hanning or the Rectangular window and the resulting FIR filter does not have ripples in the passband. Although the Blackman and the Bartlett (triangular) windows also result in filters without ripples in the passband, the Hamming window introduces the narrowest transition band for the same window size [Oppenheim and Schafer 1989; Chen 1979]. Any sinusoidal variation with a time period less than 100 ($= 1/f_cT$) timesteps is removed by the filtering process, eliminating high-frequency noise and oscillations in the waveforms. At least 100 time steps are required before one has sufficient datapoints to use this filter. For this reason, the filter is applied to the waveforms only after 100 cycles have passed. This amounts to a warmup period of 100 cycles, which speeds up the convergence check without compromising the quality of the results. The above filter parameters were chosen because they were found to work well in practice. The specific parameter values are not critical; it only matters that the filter remove the high frequency fluctuations in the waveforms.

For every state signal x_i, each of the four waveforms $P_k(x_i \mid X_0)$, $P_k(x_i \mid X_1)$, $D_k(x_i \mid X_0)$, and $D_k(x_i \mid X_1)$ is filtered separately. When all state signals have converged, the simulation is terminated and the average of the filtered versions of $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ is reported as the signal probability $P(x_i)$, for each x_i and the average of the filtered versions of $D_k(x_i \mid X_0)$ and $D_k(x_i \mid X_1)$ is reported as the transition density $D(x_i)$.

4.3 Determination of the Initial States

To carry out the simulations described above, we require two initial states X_0 and X_1 for the FSM. In case information about the design of the FSM is available, the user may supply a set of two states that correspond to normal operation of the circuit. Care should be taken to ensure that these states are far apart in the state space of the FSM, but in the same connected subset of the state space, leading to better coverage of the state space during the simulation process.

In practice, it may not be possible for the user to supply two different states. Thus, we present a simulation-based technique to determine a second state X_1, given a state X_0. If the circuit in consideration has an explicit reset state, X_0 could be chosen to be equal to that. In other cases, if no reset state is known, any state that occurs in the regular operation of the circuit could be supplied by the user to be used as X_0. In case no user-supplied information is available, we choose $X_0 = [0\ 0 \cdots 0]$ as the default value. We initiate a simulation with the FSM starting in X_0. The simulation is carried out for 100 timesteps, using either randomly selected (or user-supplied; see Section 4.5) input vectors. The required X_1 is chosen to correspond to the state with the largest Hamming distance from X_0, observed during the 100 timesteps. Although this does not guarantee optimal starting points in the state-space, we use this method in the absence of any other information about the design of the FSM.

The mechanism also ensures that X_1 corresponds to a state that the FSM would visit under the circuit's normal operation. The choice of 100 timesteps is empirical and user-controlled. This simulation, to determine X_1, is carried out before the main simulation starts and its contribution to the total run-time is negligible (less than 1%).

4.4 Low Density State Signals

We observed that, in some circuits, there exist some state signals which have very low switching activity. We refer to such signals as being *low-density* state signals. For most of these signals, the low density is observed regardless of the initial state of the FSM (X_0 or X_1). For such signals, it is typical to find that the transition density waveforms $D_k(x_i \mid X_j)$ converge, but the signal probability waveforms $P_k(x_i \mid X_j)$ may not converge as fast. This is because a signal may get stuck at logic 0 or 1 and remain there for a long time because of low switching activity. As a result the $P_k(x_i \mid X_0)$ may be close to 1 and the $P_k(x_i \mid X_1)$ waveform may be close to 0 or vice versa. Although the waveforms do not have significant variation over time and thus satisfy the *average* criterion, they do not satisfy the *difference* criterion for convergence. As a result in the case of

some circuits, the simulation process may continue for a long time, without any tangible gain of knowledge about the node statistics.

To account for these state signals we have incorporated a stopping criterion called the *low density* criterion. Essentially, we relax the error tolerance for state signals that are declared to be low density. This results in significant speedup and allows the handling of very large circuits. During the simulation, we keep note of the last timestep k_{last} in which at least one state signal was declared to have converged using the regular convergence criterion. We continue the simulation process using the regular convergence criterion as long as $k_{current} - k_{last} < k_{nochange}$, where $k_{nochange}$ is a fixed threshold value that indicates how long one is willing to wait before possibly doing some special case handling for low-density nodes. In our implementation, $k_{nochange}$ is a user-defined constant.

The special case handling is as follows. We consider all the state signals that have not yet converged and check to see if either $D_k(x_i \mid X_0)$ or $D_k(x_i \mid X_1)$ is below a user-specified low-density threshold $D_{\min}$, in which case the state signal x_i is declared to be a *low-density state signal*. Since low-density state lines will have little impact on the circuit power, all low-density signals are immediately assumed to have converged, although the user is cautioned about their presence. As obvious from the results in Section 5, the number of such state lines is very low. Since the switching activity for these state lines is low, the absolute error in the power estimate introduced as a result of this termination is negligible. The authors have also observed that the low-density criterion has no effect on the results for the remaining nodes which converge normally.

The simulation is continued further in case some state signals remain that have neither converged nor are classified as low density. If no new state signals converge in another $k_{nochange}$ timesteps, the low-density threshold is incremented by a small amount and the low-density criterion is applied again. For the benchmark circuits which we considered, this happened only once, and is pointed out in Section 5.

4.5 Input Generation

In view of Assumption 1, one requirement on the applied input sequence $U(k)$ is that it not be periodic. Another condition, required for the estimation (4) to hold, is that the different $U^{(j)}(k)$ sequences used in different simulation runs $j = 1, \ldots, N$ be selected independently. Otherwise, no limitations are placed on the input sequence.

The exact way in which the inputs are selected depends on the design and on what information is available about the inputs. For instance, if the FSM is meant to execute microcode from a fixed set of instructions, then every sequence $U^{(j)}(k)$ may be a piece of some microcode program, with each $U^{(j)}(k)$ being selected independently from some pool of typical microcode sections. This method of input generation faithfully reproduces the bit correlations in $U(k)$ as well as the temporal correlation between $U(k), U(k+1), \ldots$. Alternatively, if the user has information on the relative frequency with which instructions occur in practice, but no specific program from which to select instruction sequences, then a random number generator can be used to select instructions at random

to be applied to the machine. This would preserve the bit correlations, but not the temporal correlations between successive instructions. Conceivably, if such correlation data are available, one can bias the random generation process to reproduce these correlations.

In more general situations, where the machine inputs can be arbitrary, simpler random generation processes can be used. For instance, it may not be important in some applications to reproduce the correlations between bits and between successive vectors. The user may only have information on the statistics of the individual input bits, such as the probability $P(u_i)$ and density $D(u_i)$ for every input. In this case, one can design a random generation process to produce signals that have the required P and D statistics, as follows.

Using Equations (A.3) in Appendix A, one can compute from P and D the mean high time and mean low time of the signal. By assuming a certain distribution type for the high and low pulse widths, one can then easily generate a logic signal with the required statistics. For instance, if one uses a geometric distribution (which is equivalent to the signals x_i being individually Markov), then one obtains a fixed value for the probabilities $\mathcal{P}\{\mathbf{x_i}(k) = 1 \mid \mathbf{x_i}(k-1) = 0\}$ and $\mathcal{P}\{\mathbf{x_i}(k) = 0 \mid \mathbf{x_i}(k-1) = 1\}$, as shown in Xakellis and Najm [1994], and generates the logic signals accordingly. Incidentally, in this case, even though the inputs are Markov, the FSM itself is not necessarily a Markov system.

Finally, if only the probabilities $P(x_i)$ are available for the input nodes, and if it is not important to reproduce any input correlation information, one can generate the inputs by a sequence of coin flips using a random number generator. In this case, the inputs are said to be independent and identically distributed and the FSM can be shown to be Markov, but the individual state bits x_i may not be Markov.

Our implementation results for this approach, reported in the next section, are based on this last case of independent and identically distributed inputs. However, the technique is applicable to any other mechanism of input generation, as we have explained.

5. EXPERIMENTAL RESULTS

This technique was implemented in a prototype C program that accepts a netlist description of a synchronous sequential machine. The program performs a zero delay logic simulation and monitors the flip-flop output probabilities and densities until they converge. To improve the speed, we simulate 31 copies of the machine in parallel, using bitwise operations. We have tested the program on a number of circuits from the ISCAS-89 sequential benchmark set [Brglez et al. 1989].

All the results presented below are based on an error tolerance of 0.05 (i.e., $\epsilon = 0.05$) and 95% confidence (i.e., $\alpha = 0.05$), which implies that $N = 490$. For each circuit we choose $X_0 = [0\ 0 \cdots 0]$. Under these conditions, a typical convergence characteristic is shown in Figure 2. The two waveforms shown correspond to $p_i^{(N)}(k)$ starting from X_0 and $p_i^{(N)}(k)$ starting from X_1, for node X.3 of circuit s838.1 (this circuit has 34 inputs, 32 flip-flops, and 446 gates).

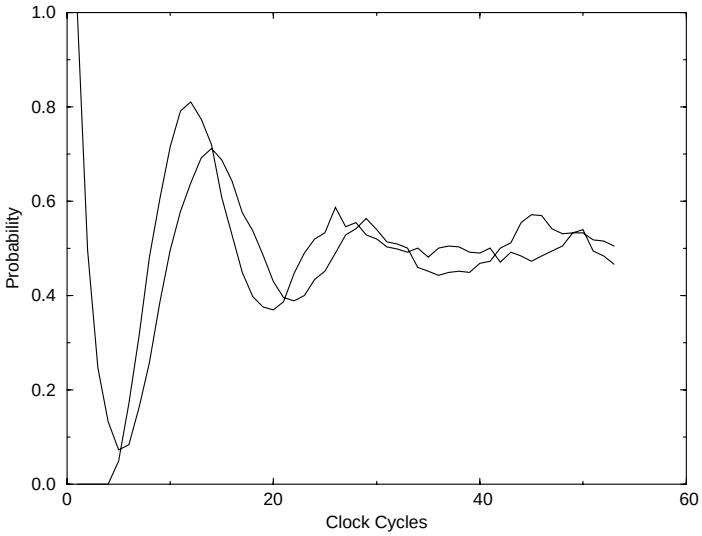

Fig. 2. Convergence of probability for s838.1, node X.3.

Table I. A Few ISCAS-89 Circuits

Circuit	No. Inputs	No. Latches	No. Gates
s1196	14	18	529
s1238	14	18	508
s713	35	19	393
s1423	17	74	657

This decaying sinusoidal convergence is typical, although in some cases the convergence is simply a decaying exponential and is much faster.

In order to assess the accuracy of the technique, we compared the (0.05, 95%) results to those of a much more accurate run of the same program. We used 0.005 error tolerance and 99% confidence for the accurate simulation which required 66,349 separate copies of the machines for each of the two initial states X_0 and X_1. Each of these 66,349 machines was simulated independently. Since we used a 100-point filter and a convergence window of 3 timesteps, at least 103 timesteps were required before we started to apply the convergence criteria. Assuming convergence in the shortest possible time (103 cycles), this implies that a total of more than 6.8 million ($103 \times 66,349$) vectors were fed into each set of machines (X_0 and X_1).

Since we are interested only in steady-state node values during the simulation, there is no need to use a more accurate timing simulator or a circuit simulator to make these comparisons. These highly accurate runs take a long time and, therefore, they were only performed on the limited set of benchmark circuits given in Table I. We then computed the difference between the statistics from the (0.05, 95%) run and those from the (0.005, 99%) run. Figure 3 shows the resulting error histogram for all the state line probability for all the flip-flop outputs from the circuits in Table I, and Figure 4 is the error histogram for the state line transition density. Notice that all the nodes have errors well within

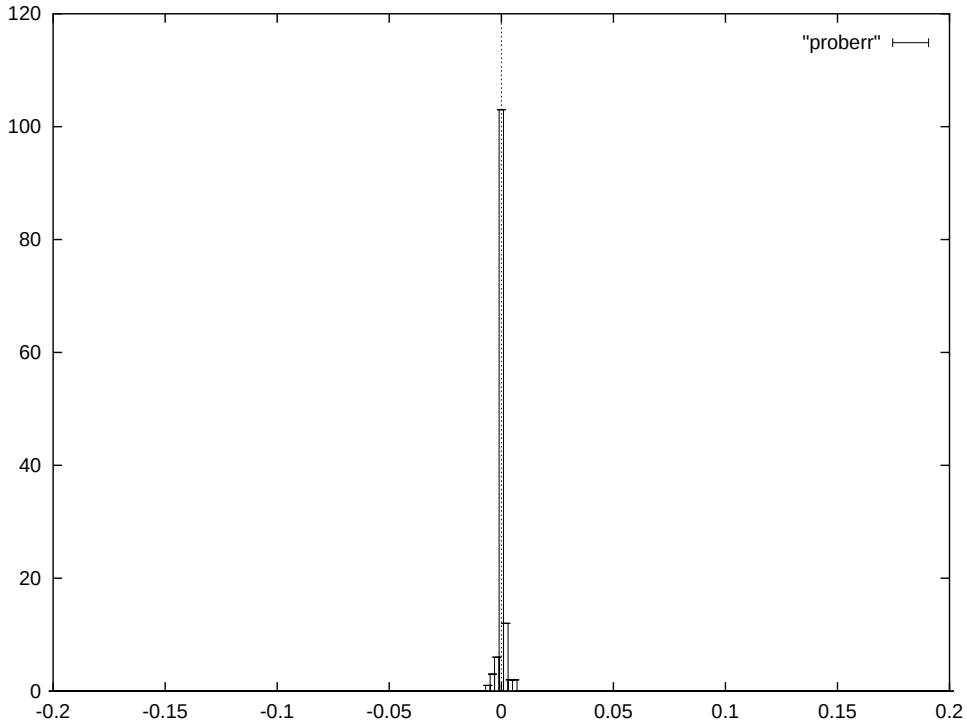

Fig. 3. Flip-flop probability error histogram.

the desired user-specified 0.05 error bounds for both the state line probability and transition density.

We monitored the speed of this technique and report some results in Table II, where the execution times are on a SUN Sparc-5 workstation. These circuits are much larger (especially in terms of flip-flop count) than the largest ISCAS-89 circuits tested in previous methods [Monteiro and Devadas 1994; Tsui et al. 1994]. Furthermore, for those circuits in the table that were also tested in Monteiro and Devadas [1994] and Tsui et al. [1994], this technique works much faster. Since our method does not use BDDs to compute probabilities, there are no memory problems with running large circuits. The largest circuit, s38584.1, requires 19.2 MB on a SUN Sparc5.

Table II also gives the number of cycles required for convergence and the number of state signals that are classified as low density. For the low-density criterion, the parameters used are: $k_{nochange} = 500$ and $D_{\min} = 0.05$ with increment (if required) of 0.05. As mentioned earlier, at least 103 timesteps are required before we start to apply the convergence criteria. Most of the smaller circuits without low-density flip-flops converge within 120 timesteps. As expected, however, the number of cycles required increases for larger circuits. The larger flip-flop count means that the machine state space is much larger and the probability of individual machine states becomes much smaller. As a result, many more cycles may be required to achieve equilibrium. Larger circuits also require more CPU time per cycle, since the simulation of the

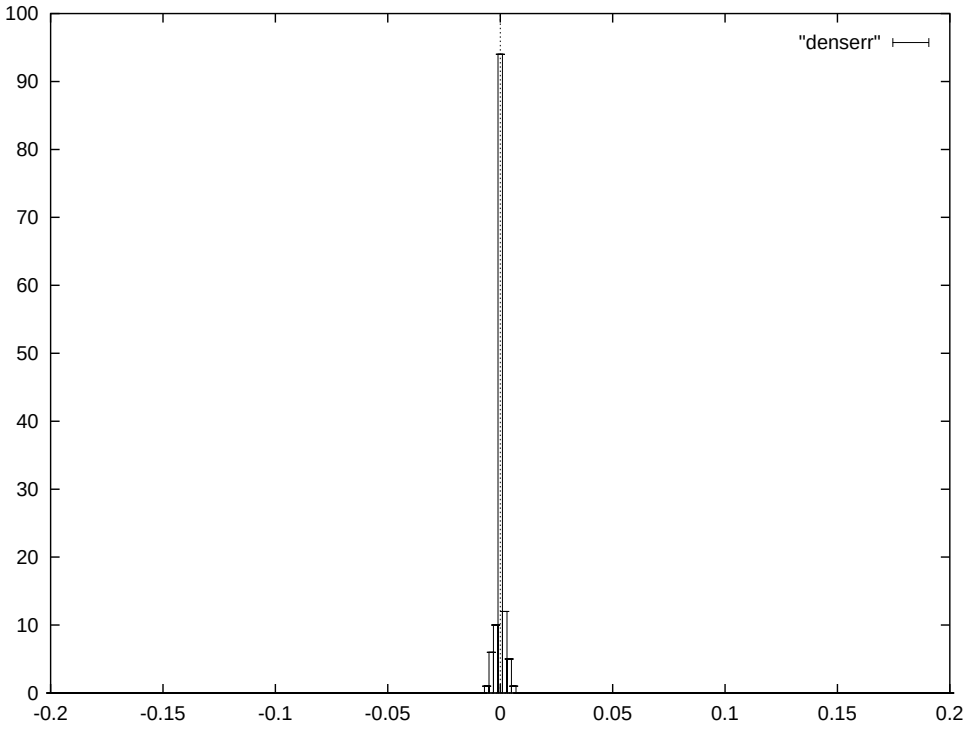

Fig. 4. Flip-flop density error histogram.

Table II. Convergence Information for ISCAS89 Benchmarks

Circuit	No. Inputs	No. Latches	No. Gates	No. Cycles	CPU Time	No. Low Dens
s208.1	10	8	104	711	36.57 sec.	1
s298	3	14	119	111	5.30 sec.	0
s344	9	15	160	112	7.21 sec.	0
s349	9	15	161	110	7.18 sec.	0
s420.1	18	16	218	711	1.14 min.	1
s444	3	21	181	111	7.38 sec.	0
s641	35	19	379	110	17.00 sec.	0
s713	35	19	393	110	17.15 sec.	0
s838.1	34	32	446	711	2.23 min.	1
s953	16	29	395	112	15.50 sec.	0
s1196	14	18	529	111	15.49 sec.	0
s1238	14	18	508	112	15.67 sec.	0
s1423	17	74	657	112	27.36 sec.	0
s1494	8	6	647	108	13.94 sec.	0
s5378	35	179	2779	179	2.67 min.	0
s9234.1	36	211	5597	614	12.12 min.	3
s13207.1	62	638	7951	1387	53.13 min.	3
s15850.1	77	534	9772	372	17.99 min.	0
s38584.1	38	1426	19253	112	10.25 min.	0

combinational part of the circuit also takes more time in comparison to smaller circuits.

In the case of circuits with low-density flip-flops, the number of cycles required was at least 603 (103 before we applied the normal criteria and 500 wait cycles before we applied the low-density criterion). Circuit s13207.1 is the only one in which we needed to increment the low-density threshold $D_{\min}$.

6. SUMMARY AND CONCLUSIONS

Most existing power estimation techniques are limited to combinational circuits, whereas all practical circuit designs are sequential. We have presented a new statistical technique for estimation of the state line statistics in synchronous sequential circuits. By simulating multiple copies of the circuit, under independently selected input sequences, statistics on the flip-flop outputs can be collected. This allows efficient power estimation for the whole design. An important advantage of this approach is that the desired accuracy of the results can be specified up front by the user; with some approximation, the algorithm iterates until the specified accuracy is achieved.

We have implemented this technique and tested it on a number of sequential circuits with up to 1526 flip-flops and a state space of size greater than 10^{459}. The additional convergence criterion for low-density nodes and the new simulation-based mechanism to determine the starting state of the FSM leads to reduced run-time. We confirm that the accuracy specified by the user is indeed achieved by our technique. The memory requirements are very reasonable, so that very large circuits can be handled with ease.

APPENDIX

A. DISCRETE-TIME LOGIC SIGNALS

Let $\mathcal{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ be the set of all integers, and let $x(k), k \in \mathcal{Z}$, be a function of discrete time that takes the values 0 or 1. We use such time functions to model discrete-time logic signals in digital circuits. The definitions and results presented below represent extensions of similar concepts developed for continuous time signals [Najm, 1993b]. The main results, Propositions 1 and 3, are therefore given without proof. In Proposition 2 we present a bounding relationship between probability and density for discrete-time signals.

A.1 Probability and Density

Notice that the set of integers $\{\lfloor -K/2 \rfloor + 1, \ldots, \lfloor +K/2 \rfloor\}$ contains exactly K elements, where $K > 0$ is a positive integer.

Definition 1. The *signal probability* of $x(k)$, denoted $P(x)$, is defined as:

$$P(x) \triangleq \lim_{K\to\infty} \frac{1}{K} \sum_{k=\lfloor -K/2 \rfloor + 1}^{\lfloor +K/2 \rfloor} x(k). \tag{A.1}$$

It can be shown that the limit in (A.1) always exists.

If $x(k) \neq x(k-1)$, we say that the signal undergoes a *transition* at time k. Corresponding to every logic signal $x(k)$, one can construct another logic signal $\mathcal{T}_x(k)$ so that $\mathcal{T}_x(k) = 1$ if $x(k)$ undergoes a transition at k; otherwise $\mathcal{T}_x(k) = 0$. Let $n_x(K)$ be the number of transitions of $x(k)$ over $\{\lfloor -K/2 \rfloor + 1, \ldots, \lfloor +K/2 \rfloor\}$. Therefore, $n_x(K) \leq K$.

Definition 2. The *transition density* of a logic signal $x(k)$, denoted by $D(x)$, is defined as

$$D(x) \triangleq \lim_{K \to \infty} \frac{n_x(K)}{K}. \tag{A.2}$$

Notice that $n_x(K) = \sum_{k=\lfloor -K/2 \rfloor + 1}^{\lfloor +K/2 \rfloor} \mathcal{T}_x(k)$, so that $D(x) = P(\mathcal{T}_x)$, and the limit in (A.2) exists.

The time between two consecutive transitions of $x(k)$ is referred to as an *intertransition time*: if $x(k)$ has a transition at i and the next transition is at $i + n$, then there is an intertransition time of length n between the two transitions. Let μ_1 (μ_0) be the average of the high (low), that is, corresponding to $x(k) = 1$ (0), intertransition times of $x(k)$. In general, there is no guarantee of the existence of μ_0 and μ_1. If the total number of transitions in positive time is finite, then we say that there is an *infinite* intertransition time following the last transition, and μ_0 or μ_1 will not exist. A similar convention is made for negative time.

PROPOSITION 1. *If μ_0 and μ_1 exist, then*

$$P(x) = \frac{\mu_1}{\mu_0 + \mu_1} \quad \textit{and} \quad D(x) = \frac{2}{\mu_0 + \mu_1}. \tag{A.3a, b}$$

PROPOSITION 2. *$P(x)$ and $D(x)$ are related as*

$$\frac{1}{2} D(x) \leq P(x) \leq 1 - \frac{1}{2} D(x).$$

PROOF. From (A.3), it is easy to arrive at:

$$\mu_1 = \frac{2P(x)}{D(x)} \quad \text{and} \quad \mu_0 = \frac{2(1 - P(x))}{D(x)} \tag{A.4}$$

Since time is discrete, then $\mu_1 \geq 1$ and $\mu_0 \geq 1$. Combining this with (A.4) leads to the required result. □

Another way of expressing this result is to say that $D(x) \leq 1 - 2|P(x) - 1/2|$, so that for a given $P(x)$, $D(x)$ is restricted to the shaded region shown in Figure 5.

A.2 The Companion Process

Let $\mathbf{x}(k)$, $k \in \mathcal{Z}$, be a discrete-time stochastic process [Najm 1993a] that takes the values 0 or 1, transitioning between them at *random* discrete transition times. Such a process is called a 0–1 *process*. A logic signal $x(k)$; can be thought of as a *sample* of a 0-1 stochastic process $\mathbf{x}(k)$; that is, $x(k)$ is one of an infinity of possible signals that comprise the family $\mathbf{x}(k)$.

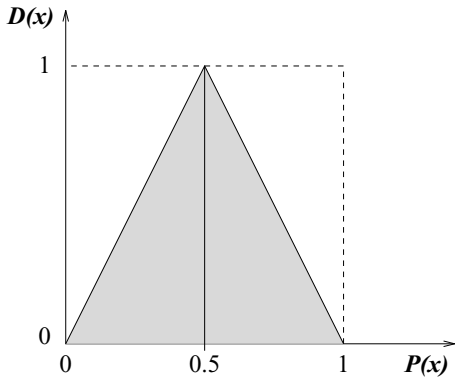

Fig. 5. Relationship between density and probability.

A stochastic process is said to be *stationary* if its statistical properties are invariant to a shift of the time origin [Najm 1993a]. Among other things, the mean $E[\mathbf{x}(k)]$ of such a process is a constant, independent of time, and is denoted by $E[\mathbf{x}]$. Let $\mathbf{n}_{\mathbf{x}}(K)$ denote the number of transitions of $\mathbf{x}(k)$ over $\{\lfloor -K/2 \rfloor + 1, \ldots, \lfloor +K/2 \rfloor\}$. For a given K, $\mathbf{n}_{\mathbf{x}}(K)$ is a random variable. If $\mathbf{x}(k)$ is stationary, then $E[\mathbf{n}_{\mathbf{x}}(K)]$ depends only on K, and is independent of the location of the time origin. Furthermore, one can show that if $\mathbf{x}(k)$ is stationary, then the mean $E[\mathbf{n}_{\mathbf{x}}(K)/K]$ is constant, irrespective of K.

Let $\mathbf{z} \in \mathcal{Z}$ be a random variable with the cumulative distribution function $F_{\mathbf{z}}(k) = 1/2$ for any finite k, and with $F_{\mathbf{z}}(-\infty) = 0$ and $F_{\mathbf{z}}(+\infty) = 1$. One might say that $\mathbf{z}$ is uniformly distributed over the whole integer set $\mathcal{Z}$. We use $\mathbf{z}$ to construct from $x(k)$ a stochastic 0–1 process $\mathbf{x}(k)$, called its *companion process*, defined as follows.

Definition 3. Given a logic signal $x(k)$ and a random variable $\mathbf{z}$, uniformly distributed over $\mathcal{Z}$, define a 0–1 stochastic process $\mathbf{x}(k)$, called the *companion process* of $x(k)$, given by

$$\mathbf{x}(k) \triangleq x(k + \mathbf{z}). \tag{A.5}$$

For any given $k = k_1$, $\mathbf{x}(k_1)$ is the random variable $x(k_1 + \mathbf{z})$, a function of the random variable $\mathbf{z}$. Intuitively, $\mathbf{x}(k)$ is a family of shifted copies of $x(k)$, each shifted by a value of the random variable $\mathbf{z}$. Thus, not only is $x(k)$ a sample of $\mathbf{x}(k)$, but one can also relate statistics of the process $\mathbf{x}(k)$ to properties of the logic signal $x(k)$, as follows.

PROPOSITION 3. *The companion process* $\mathbf{x}(k)$ *of a logic signal* $x(k)$ *is stationary, with*

$$E[\mathbf{x}] = \mathcal{P}\{\mathbf{x}(k) = 1\} = P(x) \quad \textit{and} \quad E\left[\frac{\mathbf{n}_{\mathbf{x}}(K)}{K}\right] = D(x). \tag{A.6a, b}$$

REFERENCES

BRGLEZ, F., BRYAN, D., AND KOZMINSKI, K. 1989. Combinational profiles of sequential benchmark circuits. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1929–1934.

BRODERSEN, R. W., CHANDRAKASAN, A., AND SHENG, S. 1991. Technologies for personal communications. In *Proceedings of the 1991 Symposium on VLSI Circuits* (Tokyo), 5–9.

CHEN, C. T. 1979. *One-Dimensional Digital Signal Processing*. Marcel Dekker, New York.

HACHTEL, G. D., MACII, E., PARDO, A., AND SOMENZI, F. 1994. Probabilistic analysis of large finite state machines. In *Proceedings of the 31st ACM/IEEE Design Automation Conference* (San Diego, June 6–10), 270–275.

ISMAEEL, A. A. AND BREUER, M. A. 1991. The probability of error detection in sequential circuits using random test vectors. *J. Electron. Test. 1* (Jan.), 245–256.

MILLER, I. R. AND JOHNSON, R. 1990. *Probability and Statistics for Engineers*, 4th ed., Prentice-Hall, Englewood Cliffs, N.J.

MONTEIRO, J. AND DEVADAS, S. 1994. A methodology for efficient estimation of switching activity in sequential logic circuits. In *Proceedings of the 31st ACM/IEEE Design Automation Conference* (San Diego, June 6–10), 12–17.

NAJM, F. 1993a. Statistical estimation of the signal probability in VLSI circuits. Tech. Rep. #UILU-ENG-93-2211, DAC-37, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.

NAJM, F. 1993b. Transition density : A new measure of activity in digital circuits. *IEEE Trans. Comput. Aided Des. 12*, 2 (Feb.), 310–323.

NAJM, F. 1994. A survey of power estimation techniques in VLSI circuits. *IEEE Trans. VLSI Syst. 2*, 4 (Dec.), 446–455.

NAJM, F. N., GOEL, S., AND HAJJ, I. N. 1995. Power estimation in sequential circuits. In *Proceedings of the 32nd Design Automation Conference* (San Francisco, June 12–16), 635–640.

OPPENHEIM, A. AND SCHAFER, R. 1989. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, N.J.

PAPOULIS, A. 1984. *Probability, Random Variables, and Stochastic Processes*, 2nd ed., McGraw-Hill, New York.

TSUI, C.-Y., PEDRAM, M., AND DESPAIN, A. M. 1994. Exact and approximate methods for calculating signal and transition probabilities in FSMS. In *Proceedings of the 31st ACM/IEEE Design Automation Conference* (San Diego, June 6–10), 18–23.

XAKELLIS, M. AND NAJM, F. 1994. Statistical estimation of the switching activity in digital circuits. In *Proceedings of the 31st ACM/IEEE Design Automation Conference* (San Diego, June 6–10), 728–733.

Received January 1996; revised January 1997; accepted April 2002