Power Estimation for Large Sequential Circuits

Joseph N. Kozhaya and Farid N. Najm†

Abstract – **A power estimation approach is presented in which blocks of consecutive vectors are selected at random from a user-supplied realistic input vector set and the circuit is simulated for each block starting from an unknown state. This leads to two (upper and lower) bounds on the desired power value which can be quite tight (under 10% difference between the two in many cases). As a result, the power dissipation is obtained by simulating only a fraction of the potentially very large vector set.**

Keywords – **Power estimation, sequential circuit, finite state machine (FSM)**

I. Introduction

Maximizing circuit speed and minimizing chip area used to be the only major concerns of VLSI designers. In recent years, power consumption of integrated circuits (ICs) has proved to be just as important of a concern. Thus, VLSI designs nowadays emerge as a trade-off among three goals: minimum area, maximum speed, and minimum power dissipation.

Power dissipation is a major concern of the semiconductor industry. This is because excessive power dissipation causes overheating, which may lead to soft errors or permanent damage. It also limits battery life in portable equipment. Thus, there is a need to accurately estimate the power dissipation of an IC during the design phase. We should note that by power estimation we refer to the problem of *average* power estimation. This is different from the estimation of the worst case instantaneous power. Chip reliability and equipment lifetime are directly related to the average power.

Several approaches have been proposed for power estimation [1], especially for estimation at the gate-level. However, even at the gate-level, the problem is not yet completely solved. At least two open problems remain: 1. *Accurate* and *fast* estimation of the average power dissipated by individual gates, typically inside an optimization loop, and 2. *Accurate* and *fast* estimation of the total average power dissipation in *large* sequential circuits. The words “accurate” and “fast” are emphasized in both cases to indicate that existing techniques are either inaccurate and fast or accurate and slow. The fact that the first problem is not yet solved has been clearly illustrated in [2]. In this paper, we will argue and demonstrate that the second problem is also still open, and we offer a new method which provides accurate and fast estimation of the total average power of large sequential circuits.

Since the power is pattern-dependent, the average power dissipation of a circuit is not well-defined until a specific vector set is chosen. For combinational circuits, this may not be very critical, because different vector sets may dissipate approximately the same power, provided they have approximately equal values of *switching activity*. Thus, using a set of randomly generated vectors (with the right statistics) may be appropriate for these circuits. However, this does not hold for sequential circuits, because a real vector set (as opposed to a randomly generated, artificial vector set) may contain specific vector sequences that put the circuit in specific operational modes or sub-spaces of its large state space and, in different operational modes, the circuit may dissipate quite different values of power. All one has to do is think of all the many different operational modes of a large micro-processor. Thus, for sequential circuits, the power may be *critically* dependent on the specific vector sequences that occur during typical operation.

Most existing techniques of power estimation consider simply the average switching activity and signal probability of the input signals and use either static probability propagation methods [3–6] or dynamic Monte Carlo simulation using randomly generated vectors [7, 8]. In either case, one runs the risk of taking the circuit into parts of its state space where it does not belong, i.e., into modes of operation that are unrealistic and may never be exercised in practice. When this happens, there is no guarantee that the estimated power has any relation to what the circuit will actually dissipate under typical operation.

To illustrate this problem, we have considered a number of sequential circuits and constructed two sets of input vectors for each. Both sets of vectors have the same switching activity and signal probability for each input node. However, in one vector set, the input signals were generated at random, without any corre-

† J. Kozhaya is with the ECE Dept., University of Illinois at Urbana-Champaign. F. Najm is with the ECE Dept., University of Toronto.

lation between them, and in the other non-zero correlations were considered, both in space (between pairs of bits in the same vector) and in time (between pairs of consecutive vectors). The intention is that these correlations would mimic to some degree the relationships that typically exist between signals, such as signals resulting from decoded instructions or general control signals. Note that these correlations are only the simplest kinds of correlation relations, because they do not model the temporal correlations that can exist in vector streams over *several* clock cycles. We emphasize this point to indicate that proposed approaches that use correlation coefficients [9] may be able to handle pairwise correlations between bits in a vector or between consecutive vectors, but cannot handle the variety of other input signal relations that can exist in sequential circuits. In other words, although sequence compaction methods [9] replace the *realistic, long* vector set with a smaller vector set that satisfies similar statistics, they still run the risk of taking the sequential circuit into illegal states. This is because they might introduce new vectors or vector sequences that take the sequential circuit into illegal states and thus result in wrong power estimates. Even with just these simple correlations applied, big differences are possible in the resulting power values, as shown in Table I where Pwr(uc) refers to power dissipated under the uncorrelated vector set and Pwr(co) is the power due to the correlated vector set. The error (Err) is measured as the difference between the two power values, divided by the power due to the correlated set. Note that the power in both cases (uncorrelated and correlated inputs) was measured using the same simulator, so that the errors are due only to the presence of the additional correlations in one vector set but not in the other.

TABLE I.
Differences in power values (in mW) due to presence of correlation in the vector set.

Circuit	#inputs	#latches	#gates	Power (uncorrelated)	Power (correlated)	Error (%)
s27	4	3	13	0.1405	0.1212	15.9
s298	3	14	119	0.3152	0.2615	20.5
s344	9	15	160	0.3865	0.3115	24.1
s349	9	15	161	0.3924	0.3181	23.4
s382	3	21	158	0.2021	0.1651	22.4
s386	7	6	159	0.4122	0.3591	14.8
s400	4	21	164	0.1988	0.1597	24.5
s444	3	21	181	0.2085	0.1692	23.3
s526	3	21	193	0.3319	0.2561	29.6
s526n	3	21	194	0.3375	0.2545	32.6
s641	35	19	379	0.3232	0.2982	8.4
s713	35	19	393	0.3240	0.3034	6.8
s820	18	5	289	0.4912	0.3998	22.9
s832	18	5	287	0.4679	0.3921	19.3
s1196	14	18	529	1.4719	1.7009	−13.5
s1238	14	18	508	1.5742	1.8381	−14.4
s1423	17	74	657	0.3461	0.2308	50.0
s1488	8	6	653	0.3729	0.1796	107.7
s1494	8	6	641	0.3648	0.1657	120.2
s35932	35	1728	16065	13.8721	12.2942	12.8

It would seem, therefore, that the only truly accurate power estimation method for sequential circuits is to simulate the circuit for a specific *realistic* and *typical* vector set. We refer to such a vector set as the *power vector set.* If one has a power vector set which is short enough to simulate in its entirety, then this would certainly be the method of choice. However, in practice it is very hard (almost impossible) to specify a power vector set which is both short-enough for simulation and long-enough to cover all the interesting operational modes of a large sequential circuit. Micro-processor designers will usually agree that millions of vectors may be needed in order to satisfactorily exercise their large designs.

To solve this problem, we propose a method of power estimation that takes a (potentially very long) power vector set and provides an estimate of the total power by simulating only a fraction of the vector set. The vectors to be simulated are selected by repeatedly choosing blocks of consecutive vectors at random, until certain accuracy criteria are met. We call this a *block-sampling* approach. From the repeated simulations of the blocks, we collect statistics on the *mean upper bound* and *mean lower bound* for the power per block. Using standard Monte-Carlo mean estimation techniques, the two means can be estimated with user-specified accuracy and confidence without having to simulate all blocks. The net effect is that only a fraction of the total vector set is simulated and accurate tight bounds on the total power are estimated, yielding a viable accurate power measure.

II. Problem Formulation

Let $u_1, u_2, \ldots, u_m$ be the primary input nodes of a sequential logic circuit 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 $u_i(k)$ and the *present state* values $x_i(k)$ determine the *next state* values $x_i(k+1)$ and the circuit outputs, where k denotes the clock cycle, so that the circuit implements a *finite state machine* (FSM).

Suppose a power vector set is provided which consists of the input vectors $U(1), U(2), \ldots, U(M)$, where $U(k) = [u_1(k)\ u_2(k)\ \ldots\ u_m(k)]$ is the input vector applied during cycle k, and M is the total number of vectors in the vector set. We assume that the initial state vector $X(1)$ is well-defined, so that there exists a well-defined resulting sequence of state vectors $X(1), X(2), \ldots, X(M)$, where $X(k) = [x_1(k)\ x_2(k)\ \ldots\ x_n(k)]$. The initial state need not be known, it only needs to be well-defined, i.e., not arbitrary or variable, in order for the power (due to this vector set) to be well-defined.

The total energy dissipated in the circuit in the kth cycle, denoted $e(k)$, is a function of $X(k-1)$, $X(k)$, $U(k-1)$, and $U(k)$. For $e(1)$, because $X(0)$ and $U(0)$ are not defined, we arbitrarily define $e(1) = 0$. Over a block of K consecutive input vectors, starting at cycle i, the average power dissipated is (where T is the clock period):

$$P_K(i) = \frac{1}{KT} \sum_{k=i}^{i+K-1} e(k) \tag{1}$$

If K is a constant, the same for any i, then the total power dissipation P (over the whole vector set) is given by:

$$P = \frac{1}{MT} \sum_{k=1}^{M} e(k) = \frac{1}{M} \sum_{i=-K+2}^{M} P_K(i) \tag{2}$$

The second equality is true because for any given cycle $k = k_0$, the energy $e(k_0)$ due to that cycle will occur in K different blocks and therefore will be part of K different terms $P_K(i)$. Note that for the last $K-1$ blocks, for which $i + K - 1 > M$, one should use $e(k) = 0$ in (1) for all $k = M+1, \ldots, M+K-1$. The same applies to the first $K-1$ blocks, $e(k) = 0$ for $k = -K+2, \ldots, 0$. This is required in order for the average power per block to be equal to the average power per cycle, leading to (2). If we now consider a probability experiment in which a block of vectors is chosen at random from the power vector set so that all blocks are equi-probable, then the average power per block becomes a random variable, denoted $\mathbf{P_K}$, which takes values in the set $\{P_K(-K+2), P_K(-K+3), \ldots, P_K(M)\}$. We will use bold font to denote random quantities. From (2), it becomes clear that the total power is the following *mean* or *expected value*:

$$P = E[\mathbf{P_K}] \tag{3}$$

where $E[\cdot]$ denotes the expected value operator.

If $P_K^u(i)$ and $P_K^l(i)$ are upper and lower bounds on $P_K(i)$, respectively, then we can also talk about the random variables $\mathbf{P_K^u}$ (random upper-bound value) and $\mathbf{P_K^l}$ (random lower-bound value), so that $\mathbf{P_K^l} \leq \mathbf{P_K} \leq \mathbf{P_K^u}$, which leads to:

$$E[\mathbf{P_K^l}] \leq P \leq E[\mathbf{P_K^u}] \tag{4}$$

In the next section, we propose a practical method for estimating the two bounds in (4).

III. Single Block power Analysis

If it were possible to obtain sample values of $P_K(i)$ for a sufficient number of values i, it would then be possible to estimate P based on (3) to any desired accuracy (with some specified confidence) using traditional statistical methods of mean estimation. However, since the FSM state at the start of a block is unknown, this cannot be done. Instead, our approach is based on (4) and involves using mean-estimation techniques to find two bounds on the unknown power value.

Briefly stated, we make N random choices for the block start index i (let these constitute a set of indices I) from which we compute by simulation N sample values of each of the random variables $\mathbf{P_K^u}$ and $\mathbf{P_K^l}$. We then compute the two means:

$$E[\mathbf{P_K^u}] \approx \frac{1}{N}\sum_{i\in I} P_K^u(i) \quad \text{and} \quad E[\mathbf{P_K^l}] \approx \frac{1}{N}\sum_{i\in I} P_K^l(i) \tag{5}$$

which we can use as bounds on the desired power value P, based on (4). It remains to describe how to perform the simulation in order to obtain $P_K^u(i)$ and $P_K^l(i)$, and discuss the behavior of $P_K^u(i)$ and $P_K^l(i)$ as a function of the vectors simulated. Furthermore, we need to describe how we choose values for K and N. These topics are covered below and in the next section.

A. Block Simulation

The simulation of a block of vectors is complicated by the fact that the state of the FSM at the beginning of that block is not known. Any wrong choice made for the state at that time can have the effect that the simulation of this block takes the FSM into states that never occur in practice. Therefore, we set the FSM to an all-X state (all state bits are in the unknown state) and perform three-valued gate-level simulation, with the values (0, 1, X). During the simulation of the block, we compute two bounds on the power due to that block. The upper (lower) bound is found by assuming that every signal transition containing an X value actually occurs with the X replaced by either a 0 or a 1, whichever leads to the larger (smaller) power dissipation for that transition. For instance, if the output of a gate makes an $X \rightarrow 1$ transition, then it is assumed to be a $0 \rightarrow 1$ transition for purposes of computing the upper bound and a $1 \rightarrow 1$ transition for purposes of computing the lower bound. For purposes of continuing the simulation, the transition is kept as $X \rightarrow 1$. Likewise, when the output of a gate makes an $X \rightarrow X$ transition, it is assumed to be a $0 \rightarrow 1$ (or $1 \rightarrow 0$) transition for purposes of computing the upper bound and a $0 \rightarrow 0$ (or $1 \rightarrow 1$) transition for purposes of computing the lower bound. For continuing the simulation, it is kept as an $X \rightarrow X$ transition. In this way, the true (unknown) signals in the circuit are guaranteed to be sub-sets of the simulated signals, and the true power for that block is guaranteed to be between the two resulting bounds $P_K^u(i)$ and $P_K^l(i)$.

The reason that this method can be useful in practice is that in many cases, many of the X values become definite 0 or 1 values during three-valued simulation. In fact, we have found that sufficiently many X values become known that the two bounds resulting from the simulation of one vector block can be very close, close enough to constitute a viable measure of power. A related issue of importance at this point is the initializability of circuits. A circuit is said to be *functionally initializable* if, once implemented, it can always be initialized to a definite state. On the other hand, a circuit is said to be *logically initializable* if, when started from an all-X state (unknown initial state), there exists a vector sequence that can drive it into a definite state using three-valued logic simulation.

For logically initializable circuits, we have observed that simulating a *few* vectors *typically* takes the circuit from an unknown initial state (all Xs) to a known state. The word *few* is emphasized to indicate that although the actual number of vectors varies as a function of both, the circuit and the vector stream simulated, it is typically a small fraction of the total vector stream.

We verified the above observation for *all* the circuits of the ISCAS-89 [12] benchmark circuits which are known to be logically initializable as given by [13]. This is illustrated with the histogram shown in Fig 1. We will refer to the simulation of a specific circuit for a specific vector stream as one *test case*. Also, we will refer to the number of vectors simulated before the state of the circuit becomes known as the length of an initializing sequence. The histogram shows that of the 600 test cases (20 circuits, each simulated for 30 vector streams), 573 had an initializing sequence of less than 10 vectors and only 8 required an initializing sequence larger than 50 (these eight cases, which are not shown in the figure, are: 51, 55, 59, 73, 96, 109, 109, and 267). This leads to our claim that *typically*, simulating a *logically initializable* circuit for a *few* vectors is enough to take the circuit from an unknown initial state to a known state.

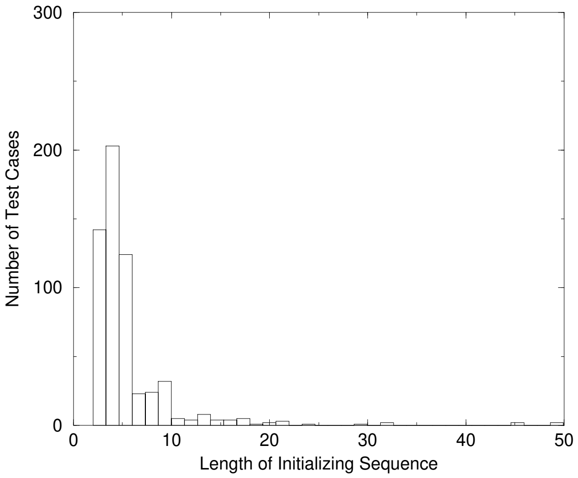

Fig. 1. Histogram for the length of an initializing sequence.

We should point out that any type of simulation model may be used - the measured power will be as accurate as the simulation model. Because we are computing the total power of the circuit (and not the powers of individual gates), we find that a logic simulator with a good timing model is sufficient. In our implementation, every gate has a scalable delay value, depending on the output loading capacitance due to its drain capacitance and the MOSFET gate capacitance of the logic gates on the fanout branches. Although three-valued logic simulation is usually associated with *zero delay* simulation, our simulator is actually a three-valued event driven *scalable delay* logic simulator. It is three-valued to account for unknown logic values since the initial state of the sequential circuit is unknown. Furthermore, it is event driven and uses scalable delay (so that different gates can have different delays), so that the estimated power includes the power due to glitches. Hence, if for two consecutive input vectors, one input of an **AND** gate is logic 1 while the other input undergoes an X $\rightarrow$ 1 transition at time t, then the output of the **AND** gate will undergo an X $\rightarrow$ 1 transition at time $t + D$ where D is the delay of the gate. Then X is assumed to be 0 or 1 for purposes of computing the bounds on the number of transitions. However, for continuing the simulation, X is maintained as an X.

Let $n_k^l(j)$ and $n_k^u(j)$ be lower and upper bounds on the number of logic transitions made by node j in clock cycle k, respectively. These are computed during the simulation by simply considering that signal transitions involving an X value can be interpreted in two ways as explained above, with one way representing more actual transitions and another way representing less. Thus, for example, upon observing a $0 \rightarrow X$ transition at node j, we would increment $n_k^u(j)$ (due to the $0 \rightarrow 1$ possibility) and not increment $n_k^l(j)$ (due to the $0 \rightarrow 0$ possibility). From this, the total energy dissipated in clock cycle k is bounded by $e^l(k) \leq e(k) \leq e^u(k)$, where the energy bounds are computed as follows:

$$e^l(k) = \frac{1}{2} \sum_j V_{dd}^2 C_j n_k^l(j) \tag{6}$$

$$e^u(k) = \frac{1}{2} \sum_j V_{dd}^2 C_j n_k^u(j) \tag{7}$$

respectively, where C_j is the node capacitance and the summations are taken over all gate/latch output nodes in the circuit. The reason for the 1/2 coefficient is that, on average, half the transitions will be low-to-high and the other half will be high-to-low. As pointed out above, one does not have to use this particular power model (6, 7), and any number of more accurate power modeling approaches can be used. All that is required is that the energy bounds $e^l(k)$ and $e^u(k)$ be computable during the simulation. In this work, this model was deemed sufficiently accurate in order to illustrate the feasibility of the approach. From this, the block upper/lower bound values $P_K^u(i)$ and $P_K^l(i)$ are computed in a way similar to (1), as follows:

$$P_K^l(i) = \frac{1}{KT} \sum_{k=i}^{i+K-1} e^l(k) \tag{8}$$

$$P_K^u(i) = \frac{1}{KT} \sum_{k=i}^{i+K-1} e^u(k) \tag{9}$$

To illustrate the process of block simulation, consider the circuit shown in Fig. 2. The circuit is simulated for the vectors shown, v_0, v_1, v_2, v_3, and v_4, starting from an unknown initial state. Note that after simulating vectors v_0 and v_1, the state of the circuit is completely known. Thus, the lower and upper bounds on the number of transitions at every node are equal for vectors v_2, v_3, and v_4. Considering node Z, this node undergoes an $X \rightarrow 0$ transition when vectors v_0 and v_1 are simulated. Thus, the lower bound on the number of transitions for this node would be 0 (assuming $0 \rightarrow 0$) while the upper bound would be 1 (assuming $1 \rightarrow 0$). Therefore, the lower and upper bounds on the power value are initially different but then start converging to the same value after the state of the circuit becomes known.

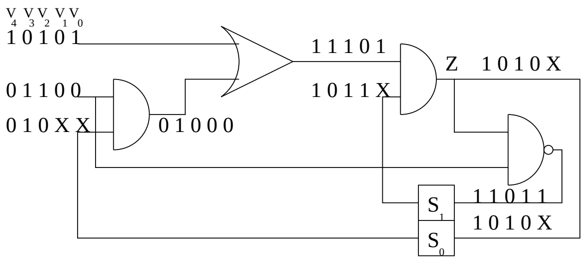

Fig. 2. Block simulation.

B. Choice of Block Size, K

The choice of block size, K, can affect the tightness of the bounds in (4). This is because the larger the block, the more probable it is that more X values will be converted to definite 0 or 1 values during the simulation. On the other hand, K should not be too large because beyond some point there will typically be very little or no reduction in the number of X values. In our implementation, K was chosen empirically, by looking at a large number of simulations, and we found that a value of $K = 500$ is appropriate.

A typical plot for a circuit with around 16000 gates is shown in Fig. 3. In practice, say for a microprocessor design, the value of K would probably have to depend on the instruction set and on the number of instructions that may be required to constitute meaningful processing tasks. In any case, the choice of K will only affect the tightness of the bounds, not their correctness.

For a circuit which is logically initializable, then (as pointed out above) in all cases that we observed, the circuit state becomes completely known after a *few* vectors (see Fig. 1). Once that happens, then the upper and lower bounds on energy per cycle ($e^u(k)$ and $e^l(k)$) become identical (equal to the true $e(k)$). Based on this, one can easily prove (see appendix) that from then on, the power bounds for that block are guaranteed to converge to the same value. This accounts for the observed tightness in our results. If the circuit is not logically initializable, then the circuit state may remain mostly unknown (most of the state bits remain X) and therefore the bounds may remain quite different, and not tight.

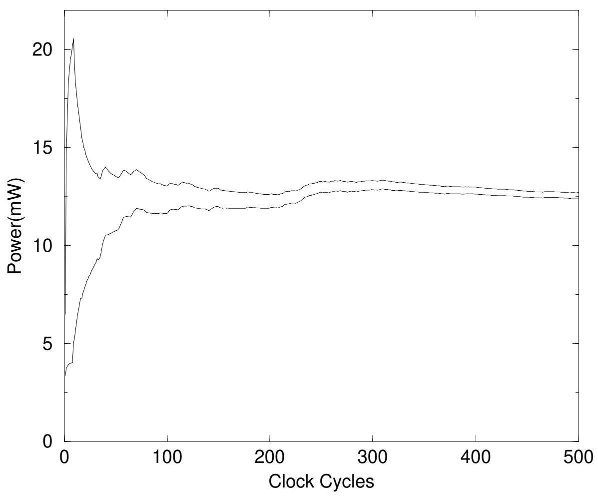

Fig. 3. Upper and lower power bounds under a correlated vector set for circuit s35932 (1728 latches and 16,065 gates).

C. Choice of Sample Size, N

The choice of sample size, N, affects the quality of the approximations in (5). It should be clear that the larger N is, the better the approximation, but how much should N be for a certain desired error tolerance? This is the classical problem of mean-estimation in statistics. We will briefly review the mean-estimation procedure with reference to an arbitrary random variable $\mathbf{x}$ whose mean $E[\mathbf{x}]$ is to be estimated from N sample values $x_1, \ldots, x_N$, using:

$$E[\mathbf{x}] \approx \mu_N = \frac{x_1 + \cdots + x_N}{N} \quad (10)$$

which is what is done in (5). Basically, $\mathbf{x}$ corresponds to the average power per block and thus, by estimating its mean, the total average power dissipated in the circuit is estimated as given in (3). In our work, the start of a block is chosen completely at random every time, independently of all prior block positions, using a uniform random number generator that gives a value between $-K+2$ and M. Hence, the values $x_1, \ldots, x_N$ are guaranteed to be samples of *independent* random variables. Furthermore, all blocks are of the same size, and thus, $x_1, \ldots, x_N$ are samples of *identically distributed* random variables.

C.1 Using the t-distribution

Therefore, $x_1, \ldots, x_N$ are samples of *independent, identically distributed (iid)* random variables. Thus, μ_N as given in (10) is a sample of a random variable called the *sample mean* [10], whose mean is equal to $E[\mathbf{x}]$ and whose variance is equal to σ^2/N, where σ^2 is the variance of $\mathbf{x}$. If the sample values $x_1, \ldots, x_N$ are taken from a *normal* population having the mean $E[\mathbf{x}]$ and the variance σ^2, then $t = \frac{\mu_N - E[\mathbf{x}]}{s_N/\sqrt{N}}$ is the value of a random variable having the t-distribution with $\nu = N-1$ degrees of freedom [11], where s_N is the standard deviation of the observed N data values $x_1, \ldots, x_N$. Consequently, with $(1-\alpha)$ confidence, it follows that [11]

$$-t_{\alpha/2} \leq \frac{\mu_N - E[\mathbf{x}]}{s_N/\sqrt{N}} \leq t_{\alpha/2} \quad (11)$$

where $0 < \alpha < 1$ and where $t_{\alpha/2}$ is defined so that the area to its right under the t-distribution curve is equal to $\alpha/2$. The value of $t_{\alpha/2}$ for a given α can be easily found using standard statistical tables. As for s_N, it is measured as follows:

$$s_N^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_N)^2 \quad (12)$$

Therefore, with confidence $(1-\alpha)$, we have:

$$\frac{|\mu_N - E[\mathbf{x}]|}{\mu_N} \leq \frac{t_{\alpha/2} s_N}{\mu_N \sqrt{N}} \quad (13)$$

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

$$\frac{s_N}{\mu_N \sqrt{N}} \leq \frac{\epsilon_1}{t_{\alpha/2}} \quad (14)$$

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

$$\frac{|\mu_N - E[\mathbf{x}]|}{\mu_N} \leq \frac{t_{\alpha/2} s_N}{\mu_N \sqrt{N}} \leq \epsilon_1 \tag{15}$$

This may also be expressed as the relative deviation from the mean $E[\mathbf{x}]$:

$$\frac{|\mu_N - E[\mathbf{x}]|}{E[\mathbf{x}]} \leq \frac{\epsilon_1}{1-\epsilon_1} = \epsilon \tag{16}$$

Here, $\epsilon > 0$ is defined as the user-specified *error tolerance*, and α (or $1-\alpha$) is the user-specified *confidence.* Thus (14) provides a stopping criterion that determines when to stop sampling in order to yield the accuracy specified in (16) with confidence $(1-\alpha)$. Notice that the required number of samples N is not known a priori, but is determined only when (14) is first met.

We should note here that it is not always the case that the sample values, $x_1, \ldots, x_N$, come from a normal population. In order to account for the case when the distribution of the random variable $\mathbf{x}$ is not normal, we make the following observation. Based on the *Central Limit Theorem* [11], the distribution of the sample mean approaches the *normal distribution* for large N. The minimum number of samples, N, to satisfy near-normality is typically about 30 [11]. Thus, if we define an $\mathcal{N}$*-sample* as $y_k = \frac{x_{30(k-1)+1} + \cdots + x_{30k}}{30}$, where $x_{30(k-1)+1}, \ldots, x_{30k}$ are samples of a non-normal distribution, then y_k is a sample of a random variable $\mathbf{y_k}$ whose distribution is near-normal. Consequently, one option in handling non-normal populations is to take the sample mean of 30 *iid* samples of the non-normal population, and consider it as one $\mathcal{N}$-sample. Then, repeat the process to obtain as many $\mathcal{N}$-samples as needed. This way, it is guaranteed that the obtained $\mathcal{N}$-samples are samples of a near-normal distribution. Hence, we can apply the previous technique of stopping the simulation when the user-specified accuracy and error-criteria are satisfied with some approximation. Note here that a minimum of 60 samples of the non-normal distribution are needed for convergence. This is because a minimum of 2 $\mathcal{N}$-samples (or sample means) are needed to satisfy the accuracy and error criteria specified in (16).

C.2 An alternative approach

To avoid the minimum requirement of 60 samples, the following approximation proves useful. For large sample sizes (N larger than 30 or so), one may approximate $\sigma/\sqrt{N}$ by $s_N/\sqrt{N}$ [11] where σ^2 is the variance of $\mathbf{x}$ and s_N is the standard deviation of the observed N data values $x_1, \ldots, x_N$, measured as given in (12). Because the distribution of the sample mean μ_N approaches the *normal* distribution for large N (30 or more), it follows that with $(1-\alpha)$ confidence [11]

$$-z_{\alpha/2} \leq \frac{\mu_N - E[\mathbf{x}]}{\sigma/\sqrt{N}} \leq z_{\alpha/2} \tag{17}$$

where $0 < \alpha < 1$ and where $z_{\alpha/2}$ is defined so that the area to its right under the standard normal distribution curve is equal to $\alpha/2$. The value of $z_{\alpha/2}$ for a given α can be easily found using standard statistical tables. In other words, we simply take N samples from the given distribution (whether normal or not). Then we simply check if (17) satisfied, replacing $\sigma/\sqrt{N}$ by $s_N/\sqrt{N}$. Hence,

$$-z_{\alpha/2} \leq \frac{\mu_N - E[\mathbf{x}]}{s_N/\sqrt{N}} \leq z_{\alpha/2} \tag{18}$$

Therefore, with confidence $(1-\alpha)$, we have:

$$\frac{|\mu_N - E[\mathbf{x}]|}{\mu_N} \leq \frac{z_{\alpha/2} s_N}{\mu_N \sqrt{N}} \tag{19}$$

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

$$\frac{s_N}{\mu_N \sqrt{N}} \leq \frac{\epsilon_1}{z_{\alpha/2}} \tag{20}$$

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

$$\frac{|\mu_N - E[\mathbf{x}]|}{\mu_N} \leq \frac{z_{\alpha/2} s_N}{\mu_N \sqrt{N}} \leq \epsilon_1 \tag{21}$$

This may also be expressed as the relative deviation from the mean $E[\mathbf{x}]$:

$$\frac{|\mu_N - E[\mathbf{x}]|}{E[\mathbf{x}]} \leq \frac{\epsilon_1}{1-\epsilon_1} = \epsilon \tag{22}$$

Here, $\epsilon > 0$ is defined as the user-specified *error tolerance*, and α (or $1-\alpha$) is the user-specified *confidence.*

We verified that replacing $\sigma/\sqrt{N}$ by $s_N/\sqrt{N}$ is indeed a valid approximation by implementing both of the above techniques and comparing the results.

The above discussion is applicable for estimating both the mean upper and lower bounds on the power dissipation, $E[\mathbf{P_K^u}]$ and $E[\mathbf{P_K^l}]$. Because the number of iterations required to estimate $E[\mathbf{P_K^u}]$ may be different from that required for $E[\mathbf{P_K^l}]$, we continue the sampling until the condition in Equation (14) has been met for both means.

IV. Experimental Results

The technique proposed above has been implemented and tested on a number of sequential benchmark circuits. All the results to be presented were performed with 5% error-tolerance ($\epsilon = 0.05$) and 95% confidence ($\alpha = 0.05$). All the circuits were derived from the ISCAS-89 benchmark circuits [12], after mapping them to a gate library with delay and capacitance values typical of 0.5μ CMOS technology. We have restricted our results to the subset of the ISCAS-89 circuits that are known to be logically initializable. This is because according to [13], almost all circuits that are functionally initializable are also logically initializable and practical circuits will always be functionally initializable. Other than this, no special considerations were used in picking the circuits below. Only two circuits were shown in [13] to be functionally but not logically initializable and, on these two circuits, our method does not work very well (meaning that although the bounds are correct, they are not tight). While more circuits may need to be tested, this may mean that the method is best suited to circuits that are known to be logically initializable.

Because no input vector sets are available for these benchmarks, we have tried to mimic the correlations that exist in real vector sets by generating a long correlated vector set consisting of 100,000 vectors. The correlation coefficients were changed arbitrarily every M vectors where M is chosen randomly. That is, randomly pick a number M_1, then generate M_1 vectors with certain correlation coefficients (signal probability, spatial and temporal correlation factors), then randomly pick another number M_2, and generate M_2 vectors with different correlation coefficients. Repeat until the whole 100,000 vectors are generated. Thus, the statistical properties of the vectors vary widely depending on where they are in the 100,000 vector stream. For each circuit, we first estimated the power due to the whole 100,000 vectors by simulation, and then used our block sampling approach to estimate the power, with 5% error-tolerance

TABLE II.
Performance under correlated input vectors. Execution time was measured on a SUN Sparc 10.

Circuit	LB (mW)	UB (mW)	Power (mW)	Tightness (%)	#cycles	Time (cpu sec)
s27	0.101191	0.101686	0.101439	0.49	15000	16.39
s298	0.357693	0.371610	0.364651	3.82	15000	129.60
s344	0.224243	0.228208	0.226226	1.75	15000	138.17
s349	0.339687	0.343261	0.341474	1.05	15000	165.53
s382	0.193261	0.198155	0.195708	2.50	15000	139.38
s386	0.285662	0.288900	0.287281	1.13	15000	207.40
s400	0.161781	0.164517	0.163149	1.68	18000	168.16
s444	0.211823	0.220167	0.215995	3.86	15000	166.59
s526	0.286471	0.299796	0.293134	4.55	15000	174.90
s526n	0.286778	0.297763	0.292271	3.76	15000	172.55
s641	0.283713	0.286737	0.285225	1.06	15000	487.67
s713	0.283585	0.288619	0.286102	1.76	15000	516.58
s820	0.273369	0.275380	0.274375	0.73	15000	411.95
s832	0.273695	0.278676	0.276185	1.80	15000	417.65
s1196	1.107530	1.108178	1.107854	0.06	15000	820.82
s1238	1.202244	1.202886	1.202565	0.05	15000	848.12
s1423	0.380139	0.386585	0.383362	1.68	17500	923.46
s1488	0.488430	0.505220	0.496825	3.38	17500	981.44
s1494	0.395587	0.407266	0.401427	2.91	15000	745.79
s35932	12.368276	13.015934	12.692105	5.10	15000	29538.44

In the first set of experiments, with results shown in Table II, we explored the tightness of the power bounds and the speed of convergence. For some details of these circuits (gate count, etc.), the reader is referred to Table I in section II. The table lists the power upper and lower bounds in mW, and the average of the two under the "Power" column. The tightness of the bounds was measured as the difference between them divided by their average, expressed as a percentage. The values illustrate that the bounds can be quite tight in most cases. The table also lists the number of cycles (i.e., vectors) that were required for convergence and the CPU time required. Note that the number of vectors required for convergence is different for different circuits. This illustrates the importance of having a convergence check (a stopping criterion). It would not be sufficient, for instance, to simulate all circuits for the same number of vectors. It is also notable that the required number of cycles is not necessarily larger for larger designs.

Then, the power estimated by our block sampling approach (average of the two bounds) was compared to that computed by simulation of the whole 100,000 vector set. The results are shown in Table III where the power values are expressed in mW. It is clear that the errors are very small and that all are below the specified 5% error tolerance.

TABLE III.
Error between simulation of all 100,000 correlated vectors and using the Block Sampling (BS) scheme.

Circuit	Power(All Vectors)	Power (BS)	error(%)	Compaction
s27	0.1016	0.1014	0.20	0.15000
s298	0.3615	0.3647	−0.89	0.15000
s344	0.2228	0.2262	−1.53	0.15000
s349	0.3420	0.3415	0.15	0.15000
s382	0.1941	0.1957	−0.82	0.15000
s386	0.2909	0.2873	1.24	0.15000
s400	0.1657	0.1631	1.57	0.18000
s444	0.2129	0.2160	−1.46	0.15000
s526	0.2855	0.2931	−2.66	0.15000
s526n	0.2891	0.2923	−1.11	0.15000
s641	0.2853	0.2852	0.04	0.15000
s713	0.2828	0.2861	−1.17	0.15000
s820	0.2774	0.2744	1.08	0.15000
s832	0.2802	0.2762	1.43	0.15000
s1196	1.1080	1.1079	0.01	0.15000
s1238	1.1933	1.2026	−0.78	0.15000
s1423	0.3948	0.3834	2.89	0.17500
s1488	0.5056	0.4968	1.74	0.17500
s1494	0.3857	0.4014	−4.07	0.15000
s35932	12.5638	12.6921	−1.02	0.15000

Table III also includes a column named "Compaction." This is the ratio of the total number of vectors simulated by the block sampling method to the total number of vectors (100,000) in the power vector set. For most of the circuits, it turns out to be enough to simulate around 15% of the total vector set. Note that this is the minimum number of cycles required for the approximations made in section III.C to be valid. The approximations hold if 30 samples or more are obtained. Thus, for our choice of block size $K = 500$, this would require the simulation of a minimum of 15000 cycles which is 15% of the total 100,000 vectors. For some of the circuits, more samples are required but note that, in all the circuits, it was enough to simulate at most 18% of the total vector set. Thus, the net effect is that the power is estimated by simulating only a small fraction of the total vector set. This feature is essential for simulation of large sequential circuits. Effectively, an *implicit compaction* of the vector set has been achieved. The adjective "implicit" denotes the fact that this was done on the fly, during the simulation, rather than up-front. We feel that this is the only correct way of performing compaction, mainly because, as observed in relation to Table II, the number of vectors required for convergence depends very much on the special characteristics of the circuit and is not determined simply by signal statistics or by circuit size. In fact, as was pointed out above, sometimes smaller circuits will require more cycles to converge.

Looking at the last column of Table III, some readers may conclude that perhaps simply simulating the first 18% (or whatever the fraction may be, according to the compaction ratio) of the vectors in the long vector set, in the order in which they occur, may be enough. This is not correct because the first section of the vector set may be biased for some reason–it may, for instance, have much lower switching activity than the rest of the vector set. The random choice of the blocks from anywhere in the vector set is essential in order to guarantee that the result is representative of all the various modes of operation in the long vector set. Granted, the random sampling does not explore *all* the vectors, but it does explore enough of them, and in the right way, in order to provide the desired result with the specified accuracy and confidence.

V. Conclusion

We have proposed a simulation-based method for estimating the power dissipation of sequential circuits. The method works by sampling blocks of consecutive vectors from a user-supplied (potentially very long) power vector set and simulating them. Because the state of the circuit at the beginning of each block is unknown, we initialize the circuit to an all-X state and simulate it for one block using three-valued logic simulation. The simulator includes delay information, so that it does capture glitching activity.

The proposed method is very efficient in providing accurate results for logically initializable circuits; that is, circuits whose state becomes known after simulating a few vectors starting from an initial unknown (all-X) state. However, if one finds that, for a given circuit, the circuit state remains unknown, then it would seem that the only fall-back position is to do a full simulation starting from a known initial state.

The major advantage of the method is that the state of the sequential circuit is always guaranteed to be valid–the FSM never goes outside its valid state space. Thus, the estimated power corresponds to realistic typical circuit operation. Another advantage of the method is that only a fraction of the vectors (around 15% for the circuits tested) in the (potentially huge) power vector set needs to be simulated.

References

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

[2] D. Brand and C. Visweswariah, "Inaccuracies in power estimation during logic synthesis," in *International Conference on Computer-Aided Design*, 1996, pp. 388–394.

[3] 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.

[4] G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Probabilistic analysis of large finite state machines," *Design Automation Conference*, pp. 270–275, June 1994.

[5] J. Monteiro and S. Devadas, "A methodology for efficient estimation of switching activity in sequential logic circuits," *Design Automation Conference*, pp. 12–17, June 6–10, 1994.

[6] C-Y Tsui, M. Pedram, and A. M. Despain, "Exact and approximate methods for calculating signal and transition probabilities in FSMs," *Design Automation Conf.*, pp. 18–23, June 6–10, 1994.

[7] F. Najm, S. Goel, and I. Hajj, "Power Estimation in Sequential Circuits," *Design Automation Conf.*, pp. 635–640, 1995.

[8] T-L. Chou and K. Roy, "Statistical estimation of sequential circuit activity," *Int'l Conf. on Computer-Aided Design*, pp. 34–37, 1995.

[9] R. Marculescu, D. Marculescu, and M. Pedram, "Sequence compaction for power estimation: theory and practice," *IEEE Transactions on Computer Aided Design*, vol. 18, no. 7, pp. 973-993, July 1999.

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

[11] I. R. Miller, J. E. Freund, and R. Johnson, *Probability and Statistics for Engineers*, 4th Edition. Englewood Cliffs, NJ: Prentice-Hall Inc., 1990, pp. 210–211.

[12] F. Brglez, D. Bryan, and K. Koźmiński, "Combinational profiles of sequential benchmark circuits," *IEEE International Symposium on Circuits and Systems*, pp. 1929–1934, 1989.

[13] J. Wehbeh, D. G. Saab, "On the Initialization of Sequential Circuits," *IEEE Int'l Test Conf.*, Altoon, PA, pp. 233–239, 1994.

Appendix

Let $P_M^l(i)$ and $P_M^u(i)$ be the lower and upper bounds obtained by simulating M vectors of a block whose starting index is i. Defining $\Delta P_M(i)$ as the difference between the upper and lower bounds and using equations (8) and (9) results in:

$$\Delta P_M(i) = P^u_M(i) - P^l_M(i) = \frac{1}{MT} \sum_{k=i}^{i+M-1} (e^u(k) - e^l(k)) \quad (A.1)$$

Similarly, let $P^l_{M+1}(i)$ and $P^u_{M+1}(i)$ be the lower and upper bounds obtained by simulating $M+1$ vectors of a block whose starting index is i. From (A.1), this leads to:

$$\Delta P_{M+1}(i) = \frac{M}{M+1} \Delta P_M(i) + \frac{e^u(i+M) - e^l(i+M)}{(M+1)T} \quad (A.2)$$

As presented in section III.A, we have observed that for logically initializable circuits, the state of the circuit is driven from an initial unknown state to a known state by simulating a few vectors. This observation was verified for *all* the circuits of the ISCAS-89 benchmark which are known to be logically initializable. Once M is larger than the few vectors required to get to a known state, then $e^u(i+M) = e^l(i+M)$ so that the last term in (A.2) becomes 0, which leads to:

$$\Delta P_{M+1}(i) < \Delta P_M(i)$$

by virtue of (A.2), since $M > 0$, so that the two bounds *converge* to the exact power value as more vectors are simulated. On the other hand, for circuits that are not logically initializable, the bounds may or may not converge, depending on the vector sequence. Note that for all circuits, the estimated bounds are *correct* upper and lower bounds of the exact power value even if they don't converge to the same value.