Accurate Power Estimation for Large Sequential Circuits[†]

Joseph N. Kozhaya and Farid N. Najm

ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

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.

1. Introduction

Power dissipation of VLSI circuits is a major concern of the semiconductor industry. Excessive power dissipation in integrated circuits causes overheating, which can 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.

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 recently 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 correlation 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. Even with just these

† This work was supported in part by the Semiconductor Research Corporation (SRC-DP-96-109), by Rockwell, and by an NSF CAREER Award (MIP-9623237).

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.
Difference in Power Values (in mW) due to Presence of Correlation in the Vector set.

Ckt	#input	#latch	#gate	Pwr(uc)	Pwr(co)	Err(%)
s298	3	14	119	0.3152	0.2615	20.54
s349	9	15	161	0.3924	0.3181	23.36
s386	7	6	159	0.4122	0.3591	14.80
s444	3	21	181	0.2085	0.1692	23.25
s526n	3	21	194	0.3375	0.2545	32.58
s641	35	19	379	0.3232	0.2982	8.38
s820	18	5	289	0.4912	0.3998	22.86
s1196	14	18	529	1.4719	1.7009	-13.46
s1488	8	6	653	0.3729	0.1796	107.63
s35932	35	1728	16065	13.8721	12.2942	12.83

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. Microprocessor 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.

2. 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. 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 $e(1)$, since $X(0)$ 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 K is a constant, the same for any i):

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

where T is the clock period, and the desired total power dissipation P (over the whole vector set) is given by:

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

Notice 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 $\{P_K(-K+2), P_K(1), P_K(2), \ldots, P_K(M)\}$. 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*:

$$P = E[\mathbf{P_K}] \qquad (3)$$

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), and it also becomes clear that:

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

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

3. Proposed approach

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 equation (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 how to choose values for K and N. These topics are covered below.

3.1 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. Therefore, we set the FSM to an all-X state (all state bits are in the unknown state) and perform 3-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 a 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. Other cases are treated similarly. 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 (more on this in section 4). 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.

Note that any kind of simulator may be used - the measured power will be as accurate as the simulation model. Since 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. Our simulator is event-driven, so that the estimated power includes the power due to glitches (i.e., hazards). Let $n_k^l(i)$ and $n_k^u(i)$ be lower and upper bounds on the number of logic transitions made by node i 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 multiple ways as explained above, including one way representing the most power per transition and another way representing the least. Thus, for example, upon observing a $0 \rightarrow X$ transition at node i, we would increment $n_k^u(i)$ (due to the $0 \rightarrow 1$ possibility) and not increment $n_k^l(i)$ (due to the $0 \rightarrow 0$ possibility). From this, the total energy dissipated in clock k is bounded by $e^l(k) \leq e(k) \leq e^u(k)$, where the energy bounds are computed as follows:

$$e^l(k) = \frac{1}{2}\sum_i V_{dd}^2 C_i n_k^l(i) \tag{6}$$

$$e^u(k) = \frac{1}{2}\sum_i V_{dd}^2 C_i n_k^u(i) \tag{7}$$

respectively, where C_i 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. From this, the block upper/lower bound values $P_K^u(i)$ and $P_K^l(i)$ are computed in a way similar to equation (1), as follows:

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

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

3.2 Choice of block size

The choice of block size, K, can affect the tightness of the bounds in (4). This is because the larger the block is, the more probable it can be 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 the development leading to equation (2), it was made clear that K should be a constant (the same for all blocks). However, during the simulation of a block of vectors, if it is found that $P_K(i)$ has already been estimated with sufficient accuracy before all K vectors have been simulated, there would clearly be no reason to continue the simulation of that block. Therefore, we use a dynamic scheme in which we may simulate only $K' < K$ vectors from a block, with K being the *hard limit* on the number of simulation vectors per block. In our implementation, this hard limit was chosen empirically, by looking at a large number of simulations, and we found that a value of $K = 500$ is appropriate. Typical plots are shown in Figs. 1 and 2. In practice, say for a micro-processor 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.

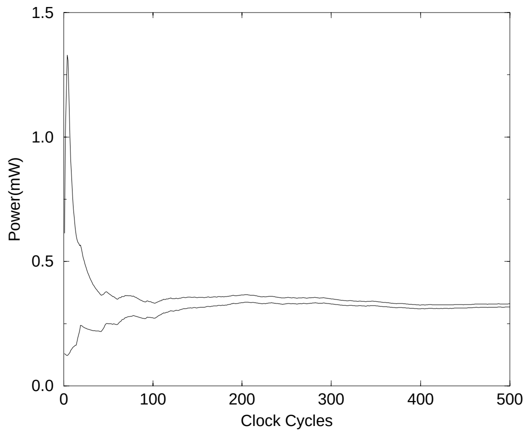

Figure 1. Upper and lower power bounds under a correlated vector set for circuit s1423 (74 latches and 657 gates).

As for the choice of K', we use the following heuristic approach to determine when to stop the simulation within a block. For every simulation cycle, we compute the latest value of the slope of the power waveform - this is a waveform which is obtained by taking the average at every time point of the two power bound waveforms. When this slope becomes small (relative to a built-in threshold), then we check the *tightness* of the bounds. The tightness is defined as the difference between the two block power bounds divided by their average. It can be shown that if the bounds on the power of every block satisfy a given tightness specification, then the two bounds on the total power in equation (4) also satisfy the same tightness. If the tightness is less than a user-specified value, then we stop the simulation of that block. Otherwise, if the tightness is not small enough but does not change for a number (in our implementation, 10) of consecutive clock cycles, then we also stop.

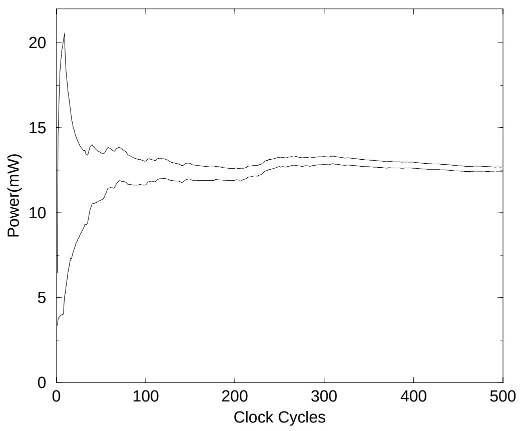

Figure 2. Upper and lower power bounds under a correlated vector set for circuit s35932 (1728 latches and 16,065 gates).

3.3 Choice of sample size

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:

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

which is what is done in (5).

In order for the results below to be true, the values $x_1, \ldots, x_n$ should be observed values of *independent, identically distributed* random variables. To guarantee this, 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. With this, the value μ_N 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 $\mathbf{x}$. Furthermore, based on the *Central Limit Theorem* [10], 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]. With $(1-\alpha)$ confidence, it then follows that [11]:

$$-z_{\alpha/2} \leq \frac{\mu_N - E[\mathbf{x}]}{\sigma/\sqrt{N}} \leq z_{\alpha/2} \tag{11}$$

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. It is also known [11] that for such large sample sizes (N larger than about 30), one may use an approximation by which $s_N/\sqrt{N}$ is used as a replacement to $\sigma/\sqrt{N}$, where s_N is the standard deviation of the observed N data values $x_1, \ldots, x_N$, measured as follows:

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

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

$$\frac{|\mu_N - E[\mathbf{x}]|}{\mu_N} \leq \frac{z_{\alpha/2}s_N}{\mu_N\sqrt{N}} \tag{13}$$

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

$$\frac{s_N}{\mu_N\sqrt{N}} \leq \frac{\epsilon_1}{z_{\alpha/2}} \tag{14}$$

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

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

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 equation (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. In our work, in computing (5), we impose a minimum sample size of 30, for the reasons explained above, so that we start checking if (14) is satisfied only after N is larger than or equal to 30.

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

4. Experimental results

The technique proposed above has been implemented and tested on a number of ISCAS-89 sequential benchmark circuits [12], after mapping them to a gate library with delay and capacitance values typical of 0.5μ CMOS technology. According to [13], *almost all* circuits that are functionally initializable are also logically initializable (with 3-valued logic simulation). Since practical circuits will always be functionally initializable, we have restricted our results to the subset of the ISCAS-89 circuits that are known to be logically 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. 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.

Since 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 100 vectors. 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 ($\epsilon = 0.05$) and 95% confidence ($\alpha = 0.05$).

Table II.
Performance Under Correlated Input Vectors. Execution Time was Measured on a sun Sparc 5.

Ckt	LB	UB	Power	Tight(%)	#cycle	Time(s)
s298	0.2928	0.3120	0.3024	6.3517	3685	18.90
s349	0.2949	0.3109	0.3029	5.2906	5540	34.67
s386	0.2883	0.3030	0.2957	4.9689	2492	24.22
s444	0.1867	0.2026	0.1946	8.1650	6178	42.10
s526n	0.2820	0.2977	0.2898	5.4316	3779	29.38
s641	0.2827	0.2972	0.2899	4.9803	2824	58.86
s820	0.3013	0.3203	0.3108	6.1218	4799	98.85
s1196	1.1151	1.1635	1.1393	4.2502	118	4.90
s1488	0.3772	0.4056	0.3914	7.2549	13085	438.95
s35932	12.3395	12.9631	12.6513	4.9293	2726	3728.69

Table II shows 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. The table lists the power upper (UB) and lower (LB) bounds in mW, and the average of the two under the "Power" column (also in mW). 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 total number of cycles (i.e., vectors) that were required for convergence and the CPU time required.

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, and the results are shown in Table III. It is clear that the errors are very small and that all are below the specified 5% error tolerance. Table III also includes a column named "Compaction." This is the ratio of the number of vectors simulated by the block sampling method, to the total number of vectors (100,000) in the power vector set. In many cases, it turns out to be enough to simulate only around 5% of the total vector set. In some cases, this ratio is larger, up to 13% in one case. 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 correct way of performing compaction, mainly because, as seen in Tables II and III, 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. Sometimes smaller circuits require more cycles to converge.

Table III.
Shows the Error Between Simulation of all 100,000 Correlated Vectors and Using the Block Sampling (bs) Scheme.

Ckt	Power(AV)	Power(BS)	err(%)	Compaction
s298	0.298088	0.302420	-1.4	0.03685
s349	0.296436	0.302960	-2.2	0.0554
s386	0.293828	0.295707	-0.6	0.02492
s444	0.188459	0.194680	-3.3	0.06178
s526n	0.285039	0.289894	-1.77	0.03779
s641	0.288030	0.289987	-4.1	0.02824
s820	0.304192	0.310878	-2.2	0.04799
s1196	1.151242	1.139382	1.0	0.00118
s1488	0.387748	0.391459	-3.5	0.13085
s35932	12.454792	12.651351	-1.6	0.02726

5. 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. Since the state of the circuit at the beginning of each block is unknown, we put the circuit in 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. During the simulation of each block, we compute an upper bound and a lower bound on the power due to that block, which we found can be very tight. The blocks are selected at random from anywhere in the power vector set, and the process is repeated until, using statistical methods of mean estimation, the upper and lower bounds on the power have been determined with sufficient (user-specified) accuracy and confidence.

A 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 in the (potentially huge) power vector set need to be simulated. For the 100,000 vector sets that we considered, this fraction varied between 0.1% and 13%.

References

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

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

[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 Conf.*, 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 Conf.*, 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, "Efficient power estimation for highly correlated input streams," *Design Automation Conf.*, pp. 628–634, June 12–16, 1995.

[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.