Sequence Compaction for Power Estimation: Theory and Practice

Radu Marculescu, *Member, IEEE*, Diana Marculescu, *Member, IEEE*, and Massoud Pedram, *Senior Member, IEEE*

***Abstract*—Power estimation has become a critical step in the design of today's integrated circuits (IC's). Power dissipation is strongly input pattern dependent and, hence, to obtain accurate power values one has to simulate the circuit with a large number of vectors that typify the application data. The goal of this paper is to present an effective and robust technique for compacting large sequences of input vectors into much smaller ones such that the power estimates are as accurate as possible and the simulation time is reduced by orders of magnitude. Specifically, this paper introduces the hierarchical modeling of Markov chains as a flexible framework for capturing not only complex spatiotemporal correlations, but also dynamic changes in the sequence characteristics. In addition to this, we introduce and characterize a family of variable-order dynamic Markov models which provide an effective way for accurate modeling of external input sequences that affect the behavior of finite state machines. The new framework is very effective and has a high degree of adaptability. As the experimental results show, large compaction ratios of orders of magnitude can be obtained without significant loss in accuracy (less than 5% on average) for power estimates.**

***Index Terms*— Dynamic Markov modeling, hierarchical Markov modeling, Markov sources, power estimation, vector compaction.**

I. Introduction

A. Basic Issues and Prior Work

COMPUTER-AIDED design (CAD) tools play a significant role in the efficient design of the high-performance digital systems. In the past, time, area, and testability were the primary concerns of the CAD community during the optimization phase. With the growing need for low-power electronic circuits and systems, power analysis and low-power synthesis have become crucial tasks that must be also addressed.

Having a gate-level implementation of the target circuit, to estimate the dynamic power consumption, we have to sum over all gates the average power dissipation due to the capacitive switching currents; that is, $P_{avg} = (f_{\text{clk}}/2) \cdot V_{DD}^2 \cdot \sum_g (C_g \cdot sw_g)$, where f_{clk} is the clock frequency, V_{DD} is the supply voltage, C_g and sw_g are the capacitance and the average switching activity at the output of gate g, respectively. As we can see, the average switching activity of every gate in the circuit is a key parameter that needs to be correctly determined, particularly if the node-by-node power estimation is of interest. Since most of the power consumption for digital circuits mapped with standard libraries comes actually from the output load charging and discharging, throughout our presentation we will neglect the internal power dissipation. This is in agreement with the vast majority of work of other researchers who also considered only external capacitance charging and discharging in their power models.

Existing techniques for power estimation at gate- and circuit-level can be divided in two main classes: dynamic and static [1], [25]. *Dynamic techniques* [2], [3] explicitly simulate the circuit under a "typical" input stream. Consequently, their results depend on the simulated sequence, and the required number of simulated vectors is usually high. These techniques can provide sufficient accuracy at the expense of large running times. Switching activity information can be extracted by doing exhaustive simulation on small circuits; it is, however, unrealistic to rely on simulation results for large circuits. To address this problem, a Monte Carlo simulation technique was proposed in [4]. This technique uses an input model based on a Markov process to generate the input stream for simulation. The approach has two deficiencies. First, the required number of samples, which directly impacts the simulation run time, is approximately proportional to the ratio between the sample variance and the square of the sample mean value. For certain sequences, this ratio becomes large, thus significantly increasing the simulation run time. Second, if the sample distribution significantly deviates from the normal distribution, the simulation may terminate prematurely. Difficult distributions that cause premature termination are bimodal, multimodal, and distributions with long or asymmetric tails [5]. The efficiency of the existing statistical techniques for power estimation in sequential circuits is even lower than that for combinational circuits [6], [7].

Static techniques rely on probabilistic information about the input stream (e.g., switching activity of the inputs, signal correlations, etc.) to estimate the internal switching activity of the circuit. These techniques generally provide sufficient accuracy with low computational overhead. However, they cannot accurately capture factors such as slew rates, glitch generation, and propagation. In addition, a major challenge in probabilistic power estimation approaches is the ability to

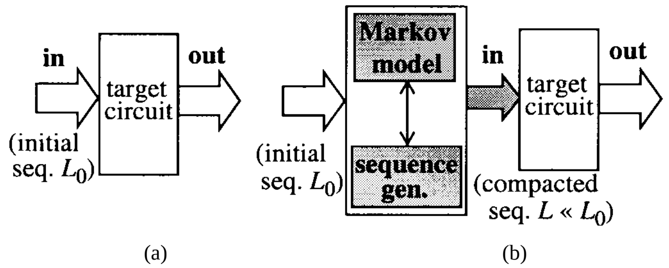
Manuscript received March 17, 1998; revised October 30, 1998. This work was supported by DARPA under Contract F33615-95-C1627 and the National Science Foundation (NSF) under Contract MIP-9628999. This paper was recommended by Associate Editor E. Macii.

R. Marculescu is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA.

D. Marculescu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA.

M. Pedram is with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089 USA.

Publisher Item Identifier S 0278-0070(99)05035-6.



(a) (b)

Fig. 1. Data compaction for power estimation.

account for internal dependencies due to the reconvergent fanout in the target circuit. This problem, which we will refer to as the "*circuit problem*," is by no means trivial. Indeed, a whole set of solutions have been proposed, ranging from approaches which build the global ordered binary decision diagrams (OBDD's) [8] and, therefore, capture all internal dependencies, to efficient techniques which partially account for dependencies in an incremental manner [9]–[12].

The authors have pointed out the importance of correlations not only inside the target circuit, but also at its primary inputs [13]. We will refer to this issue as the "*input problem*" and mention that it is important not only in power estimation, but also in low-power design. Generating a minimal-length sequence of input vectors which satisfies some prescribed statistics is a nontrivial task. The reason is that the input statistics that must be preserved or reproduced during sequence generation may be quite complex. On the other hand, it is impractical to simulate large circuits using millions or even tens of thousands of input vectors and, therefore, the length of the simulation sequence is another important issue that must be considered.

The research presented in this paper shifts the focus from the "*circuit problem*" to the "*input problem*" and proposes an original solution for power estimation based on the paradigm of *sequence compaction*. This kind of technique is appealing because it is practically independent of the actual implementation of the target circuit. It can, therefore, be used early in the design cycle when the structure of the circuit has not been determined yet. The basic idea is illustrated in Fig. 1. To evaluate the total power consumption of a target circuit for a given input sequence L_0 [Fig. 1(a)], we first derive the Markov model of the input sequence and then, having this compact representation, we generate a much shorter sequence L, equivalent with L_0, which can be used with any available simulator to derive accurate power estimates [Fig. 1(b)].

The key element in this schema is the actual Markov model used to represent the initial input sequence. In [14] and [15], preliminary efforts in using this methodology have been presented. As the experimental results show, for homogeneous input sequences, these approaches perform very well. Large compaction ratios of 1–3 orders of magnitude have been obtained without significant loss (less than 5% on average) in the accuracy of power estimates. However, for input sequences that exhibit widely different transition behaviors over time, the overall accuracy can suffer because the probabilistic model used in [14] and [15] is a *flat* model; that is, it models the *average* behavior of the input sequence, but does not adapt very well to changes in the input characteristics. For real sequences which may contain a mixture of stimuli with very different switching activities, a compaction technique with higher adaptability is clearly needed. In addition to this, the model considered in [14] and [15] is based only on a first-order Markov chain. As it will be shown later in this paper, this is not sufficient for sequence compaction for power estimation in finite state machines (FSM's). Temporal correlations longer than one clock cycle may affect the overall behavior of the FSM and, therefore, result in very different power consumptions.

In what follows, we will address these two issues and provide a new framework for sequence compaction which can be successfully applied to both combinational and sequential circuits.

B. Overview of the New Approach

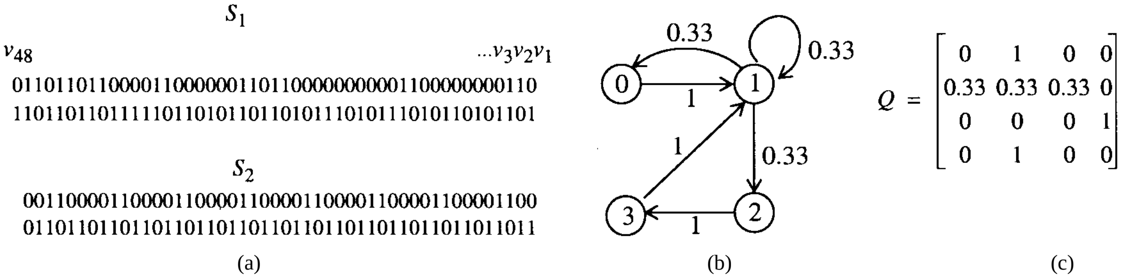
The foundation of the new approach is probabilistic in nature; it relies on *adaptive* (*dynamic*) *modeling* of binary input streams as Markov sources of information. The adaptive modeling technique itself (known in the data compression literature as *dynamic Markov chain* or *DMC modeling* [16]) was recently used in power estimation [15]. However, the model in [15] is not completely satisfactory for our purposes. In this paper, we thus extend the initial formulation to capture not only correlations between successive input patterns, but also temporal dependencies of higher orders by using *dynamic Markov trees of order* k (DMT$_k$). We also provide an original solution to distinguish between subsequences with different transition behaviors by structuring the input space as a multilevel stochastic process called *hierarchical Markov model*.

As a final note, we mention that for both combinational and sequential compaction, by using the DMC modeling technique that we propose, we do not produce new vectors; that is, all patterns that occur in the final compacted sequence, are also present in the original one. This is a fundamental theoretical difference compared to the case when new vectors are allowed in the final sequence [17], [24]. Because the search space is much larger in the latter case, our problem of sequence compaction is more constrained compared to the case of producing shorter sequences when new vectors are allowed.

In summary, both simulation-based and analytic techniques for power estimation may benefit from this research. The issues being raised are new and represent an important step toward reducing the gap between the simulative and probabilistic techniques commonly used in power estimation.

C. Organization of the Paper

The remainder of this paper is organized into four main sections. Section II reviews the background necessary to understand the proposed methodology. Sections III and IV present in detail the probabilistic models for sequence compaction which are applicable to combinational and sequential circuits, respectively, and the experimental results obtained on common benchmark circuits. Finally, in Section V, we summarize our major contribution.



$$Q = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.33 & 0.33 & 0.33 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

(a) (b) (c)

Fig. 2. Sequence characterization with STG and transition matrix.

II. Background on Finite-Order Markov Chains

In this section we present the basic definitions and notations that will be used in the paper. Far from being exhaustive, we restrict our attention to only those concepts that are required by our presentation. For a complete documentation, the reader is referred to monographs [18], [19].

A *stochastic process* is defined as a family of random variables $\{x(t), t \in T\}$ defined on a given probability space and indexed by the parameter t, where t varies over the index set T. The stochastic process is said to be *stationary* when it is invariant under an arbitrary shift of the time origin. In this case, the values assumed by the random variable $x(t)$ are called *states,* and the set of all possible states forms the *state-space* of the process.

A *Markov process* $\{x(t), t \in T\}$ is a stochastic process whose future evolution depends only on its current state and not on its past. This is the so called "Markov property" and defines a fundamental subclass of stochastic processes. We shall assume that the transitions out of state $x(t)$ are independent of time and, in this case, the Markov process is said to be *time-homogeneous*.

If the state-space of a Markov process is *discrete,* the Markov process is referred to as a Markov chain (MC). In the following, we consider only MC's with finite state-space. If we assume that the index set T is also discrete, then we have a *discrete-parameter* MC. We may assume without loss of generality that $T = \{0, 1, 2, \cdots\}$ and denote a generic MC as $\{x_n\}_{n \geq 0}$.

Definition 1—Lag-One MC: A discrete stochastic process $\{x_n\}_{n \geq 0}$ is said to be a lag-one MC if, at any time step $n \geq 1$ and for all states x_n, the following holds:

$$\begin{aligned} &p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1}, x_{n-2} = \alpha_{n-2}, \cdots, x_0 = \alpha_0) \\ &\quad = p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1}). \end{aligned} \tag{1}$$

The conditional probabilities $p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1})$ are called *single-step transition probabilities* and represent the conditional probabilities of making a transition, at time step n, from state x_{n-1} to state x_n. In homogeneous MC's these probabilities are independent of n and consequently written as $p_{ij} = p(x_n = j | x_{n-1} = i)$, for all $n = 1, 2, \cdots$. The matrix Q, formed by placing p_{ij} in row i and column j, for all i and j, is called the *transition probability matrix*. We note that Q is a *stochastic matrix* because its elements satisfy the following two properties: $0 \leq p_{ij} \leq 1$ and $\sum_j p_{ij} = 1$.

An equivalent description of the MC can be given in terms of its *state transition graph* (STG). Each node in the STG represents a state in the MC, and an edge labeled p_{ij} (from node i to node j) implies that the one-step transition probability from state i to state j is p_{ij}.

Example 1: Let S_1 and S_2 be two 2-bit sequences, of length 48, as shown in Fig. 2(a). These two sequences, have exactly the same set of first-order temporal statistics that is, they cannot be distinguished as far as wordwise one-step transition probabilities are concerned. In fact, in Fig. 2(b) we provide the wordwise transition graph for these two sequences. Each node in this graph is associated to a distinct pattern that occurs in S_1 and S_2 (the upmost bit is the most significant one, e.g., in S_1, v_1 = "1," v_2 = "2," v_3 = "3," $\cdots$, v_{48} = "1"). Each edge represents a valid transition between any two valid patterns and has a nonzero probability associated with it. For instance, the pattern "3" in S_1 and S_2 is always followed by "1" (thus the edge between nodes "3" and "1" has the probability 1), whereas it is equally likely to have either "0," "2," or "1" after pattern "1." Starting with different initial states and using a random number generator we may, of course, generate other sequences equivalent with S_1 and S_2 as far as the one-step transition probabilities are concerned. We can then see the graph in Fig. 2(b) as a compact, canonical, characterization of sequences S_1 and S_2. Suppose now that we want to compute the occurrence probability of the string v = "01 10" that is, the probability that the transition $1 \rightarrow 2$ is taking place in S_1. To this effect, we just use $p(v) = p(v_1v_2) = p(v_1) \cdot p(v_2|v_1)$, which gives the value of 1/6. If we are interested in finding the two-step transition probability $0 \rightarrow 1 \rightarrow 1$ in S_2, then using $p(v) = p(v_1v_2v_3) = p(v_1) \cdot p(v_2|v_1) \cdot p(v_3|v_2v_1)$, we get the value of 1/6. The matricial representation, equivalent with the STG, is given in Fig. 2(c). We can easily verify that the sum of all elements on each row is 1, Q thus being indeed a stochastic matrix.

Definition 2—Recurrent State: A state in the MC is called *recurrent* if the probability of returning in this state after $n \geq 1$ steps is greater than zero. Otherwise, the state is called *transient.* If the greatest common divisor over all such integers n is $d = 1$, then the state is also called *aperiodic*.

In our subsequent discussion, we will consider that *all states* are *recurrent* since all transient states vanish after a small number of steps.

Definition 3—Nondecomposable MC: A Markov chain is said to be *nondecomposable* if every state can be reached from every other state in a finite number of steps.

Note: The STG in Fig. 2(b) is nondecomposable; we note that, in general, the STG associated to any input sequence of vectors is also nondecomposable.

Changes of states over $n > 1$ time steps are given by probability rules simply expressed in terms of p_{ij}. Let us denote by p_{ij}^{n} the probability of transition from state i to state j in exactly n steps, namely: $p_{ij}^{n} = p(x_{m+n} = j | x_m = i)$, whatever the integer m. It can be easily seen that probabilities p_{ij}^{n} (which are called *n-step transition probabilities*) represent the entries of the Q^n matrix (called *n-step transition matrix*), $n \geq 1$. The Q^n matrix itself is still a stochastic matrix and satisfies the identity $Q^{m+n} = Q^m \cdot Q^n$, $m, n \geq 0$ (Q^0 is by definition the unit matrix I) or just the system of equations $p_{ij}^{m+n} = \sum_k p_{ik}^{m} \cdot p_{kj}^{n}$, known as the *Chapman–Kolmogorov* equations [18]. In other words, to go from i to j in exactly $(m + n)$ steps, it is necessary to go first from i to an intermediate state k, in m steps, and then from k to j in the remaining n steps. By summing over all possible intermediate states k, we consider all possible distinct paths leading from i to j in $(m + n)$ steps. Assuming stationarity, if $\pi = [\pi_i]$ denotes the *state probability* vector of the MC, then from Chapman–Kolmogorov equations we have that $\pi \cdot Q = \pi$.

Theorem 1—[19]: For a nondecomposable MC, the equation $\pi \cdot Q = \pi$ has a unique solution that represents the *stationary distribution* of the MC. □

Note: If the Markov chain is aperiodic, then Q^n converges to a stable matrix when $n \to \infty$, and π can be found to be any row of the limiting matrix.

Definition 4—Lag-k MC: A discrete stochastic process $\{x_n\}_{n \geq 0}$ is said to be a *lag-k MC* if, at any time step $n \geq k$, we have

$$\begin{aligned} p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1}, x_{n-2} = \alpha_{n-2}, \cdots, x_0 = \alpha_0) \\ = p(x_n = \alpha_n | x_{n-1} = \alpha_{n-1}, x_{n-2} = \alpha_{n-2}, \cdots, \\ x_{n-k} = \alpha_k). \end{aligned} \tag{2}$$

It should be noted that any lag-k MC can be reduced to a lag-one MC based on the following result.

Proposition 1—[19]: If $\{u_n\}_{n \geq 0}$ is a lag-k MC then $\{v_n\}_{n \geq 0}$ where $v_n = (u_n, u_{n+1}, \cdots, u_{n+k-1})$, is a multivariate first-order MC. □

Consequently, the study of lag-k MC's is practically reduced to the study of the properties satisfied by lag-one MC's. We will subsequently refer mostly to lag-one MC's but, by virtue of Proposition 1, all results easily translate to lag-k MC's.

III. Sequence Compaction for Combinational Circuits

In what follows, we will use elements from the theory of discrete-parameter time-homogeneous MC's to derive a probabilistic model for sequence compaction for power estimation in combinational circuits.

A. Problem Formulation

As shown in Fig. 3, we model the "tuple" (*input_sequence, target_circuit*) by the "tuple" (*Markov_chain, target_circuit*), where *Markov_chain* represents the Markov chain that models the *input_sequence* and *target_circuit* is the combinational circuit where power consumption has to be determined. Let x_n denote a random variable associated to primary inputs of the circuit shown in Fig. 3; $p(x_n)$ is then the probability that the input is x_n at time step n. We are interested in defining the probabilities $p(x_n)$ and $p(x_n x_{n-1})$ because they completely capture the characteristics of the input that feeds the target circuit. Using these probabilities, the vector compaction problem can be formulated as follows.

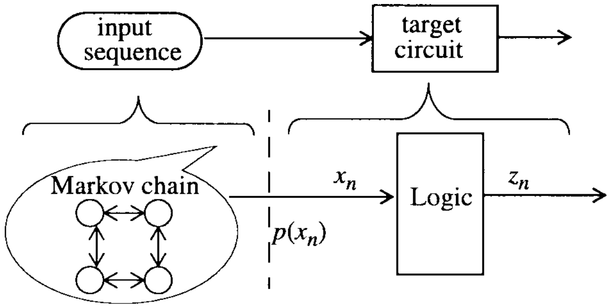


Fig. 3. The tuple (*Markov-Chain, target_circuit*).

Problem Formulation: For a sequence of length L_0, find another sequence of length $L < L_0$ (consisting of a subset of the initial sequence), such that the *average transition probability* on the primary inputs is preserved *wordwise*, for two consecutive time steps. More formally, the following condition should be satisfied:

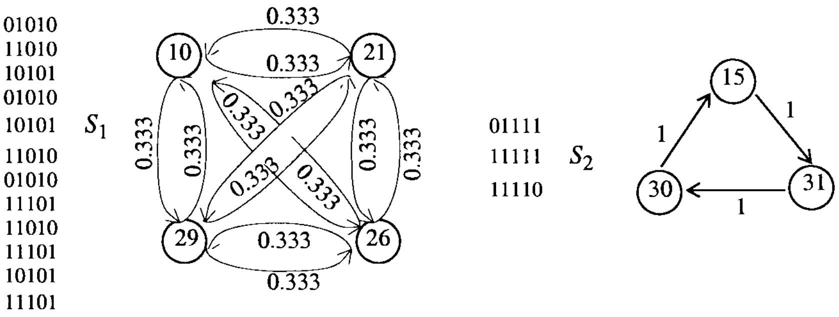
$$|p(x_n x_{n-1}) - p^*(x_n x_{n-1})| < \varepsilon \tag{3}$$

where p and p^* are the probabilities in the original and compacted sequences, respectively, and ε is an infinitesimal quantity. □

This condition simply requires that the joint transition probability for the primary inputs is preserved within a given level of error, for any two consecutive vectors. We want to prove now that indeed, by having satisfied relation (3), it is guaranteed to produce a new sequence which is asymptotically close to the original one as far as the total power consumption in the target circuit is concerned. The proof will involve several intermediate results as shown subsequently.

Proof of Correctness: As stated in the previous section, Q represents the stochastic matrix associated to the original input sequence, i.e., $p_{ij} = p(v_j | v_i)$, where v_i, v_j are any two consecutive vectors. To produce an equivalent sequence from a reference one, one should preserve the word-level transition probabilities. This essentially becomes the problem of preserving both conditional and state probabilities because $p_{i \to j} = \pi_i \cdot p_{ij}$, where π_i is the ith component of the state probability vector and $p_{i \to j}$ represents the transition probability of going from vector v_i to vector v_j. (From Theorem 1, it can be seen that π is the left eigenvector that corresponds to the eigenvalue $\lambda = 1$ in the general equation $\pi \cdot Q = \lambda \cdot \pi$.) At this point, we emphasize the importance of stationarity condition for defining the state probability vector π_i.

To complete the proof, we note that every stochastic matrix has one as simple eigenvalue and all other eigenvalues have absolute values less than one. (This is in fact a consequence



(a) (b)

Fig. 4. Two sequences and their corresponding transition graphs.

of the Perron–Frobenius theorem [20] which states that for every matrix with nonnegative entries, there exists a simple,[1] positive eigenvalue greater than the absolute value of any other eigenvalue.) This result is very important because it makes possible to analyze the effect of perturbation of matrix Q on the eigenvectors that correspond to the eigenvalue one. To this end, let us assume that the newly generated sequence is characterized by the matrix $Q^* = [p^*_{ij}]$ where $p^*_{ij} = p_{ij} + \varepsilon_{ij}$ (ε_{ij} represents the error induced by perturbations) and $|\varepsilon_{ij}| < 1$. We can write $Q^* = Q + \varepsilon \cdot B$ where $\varepsilon = \max|\varepsilon_{ij}|$ and $b_{ij} = \varepsilon_{ij}/\varepsilon$. Because Q^* characterizes a sequence of vectors, it is also a stochastic matrix and, therefore, it has an eigenvalue $\lambda^* = 1$. But, from the theory of algebraic functions [21], for any eigenvector π of Q corresponding to the simple eigenvalue $\lambda = 1$, there exists an eigenvector π^* of Q^* corresponding to the simple eigenvalue $\lambda^* = 1$, such that $||\pi - \pi^*|| = 0(\varepsilon)$ (read as "*zero of epsilon*"), where $0(\varepsilon)$ is any power series in ε (convergent for sufficiently small ε) having the form $k_1\varepsilon + k_2\varepsilon^2 + \cdots$. As a consequence, since $|p_{ij} - p^*_{ij}| = 0(\varepsilon)$, it is easy to see that $|p_{i\to j} - p^*_{i\to j}| = 0(\varepsilon)$. ■

Summarizing, we have that:

Corollary 1: If the stochastic matrix Q is properly preserved during the compaction process, then the transition probabilities of the newly generated sequence are *asymptotically close* to the original ones, that is $|p_{i\to j} - p^*_{i\to j}| = 0(\varepsilon)$. □

We have, thus, proved that we can *asymptotically* reproduce an initial sequence by preserving its matrix Q. From a practical point of view, let us see the implications of the above corollary on total power consumption in a target circuit when the original input sequence is approximated by a new one.

Corollary 2: If P and P^* are the values of the total power consumption which are obtained for two sequences satisfying the conditions in Corollary 1, then we have that $|P - P^*| = 0(\varepsilon)$. □

Proof: We have that $P = (f_{\text{clk}}/2) \cdot V^2_{DD} \cdot \sum_{i,\,j,\,k} p_{i\to j} \cdot C_k \cdot n^k_{ij}$, where C_k is the output capacitance of gate k and n^k_{ij} is the number of transitions at the output of gate k when vector v_i, followed by vector v_j, is applied at the input of the circuit. Assuming that the input sequence is approximated by another input sequence such that the new set of transition probabilities satisfies $|p_{i\to j} - p^*_{i\to j}| = 0(\varepsilon)$, then the error made in the value of total power consumption is given by

$$|P - P^*| \le \frac{f_{\text{clk}}}{2} \cdot V^2_{DD} \cdot \sum_{i,\,j,\,k} |p_{i\to j} - p^*_{i\to j}| \cdot C_k \cdot n^k_{ij} = 0(\varepsilon). \qquad ■$$

Corollary 2 basically shows that, if the new sequence is asymptotically close to the original one, then the same type of asymptotic relationship holds for the total power values. We have, therefore, proved that a first-order probabilistic model is *sufficient* to perform sequence compaction for power estimation in combinational circuits. The remaining portion of this section describes how we can efficiently do compaction in practice.

B. Hierarchical Models

This section introduces the hierarchical modeling of Markov chains as a flexible framework for capturing not only complex spatiotemporal correlations, but also the dynamic changes in the sequence characteristics.

1) Nonhomogeneous Sequences: In [14] and [15], the authors present preliminary results in solving the vector compaction problem when only first-order temporal correlations are taken into account. The Markov model used to represent the initial input sequence is a *flat* model; that is, it models only the *average* behavior of the input sequence. The primary disadvantage of any flat model is the fact that it does not adapt very well to changes in the input characteristics. To illustrate the significance of this issue, we consider the following example.

Example 2: Let S_1 be a 5-bit sequence as shown in Fig. 4(a); next to it, we represent the word-level transition graph that corresponds to this sequence.[2] This particular set of inputs behaves like a pseudorandom sequence because any vector is equally likely to be followed by any other remaining pattern. In Fig. 4(b) we consider another sequence S_2, which is completely deterministic and highly correlated. It has an average value of 1.33 transitions per step, thus producing less activity compared to S_1.

Suppose that we duplicate 25 times the original sequence S_1 and 100 times the sequence S_2, getting two new sequences

[1] This means that the multiplicity of the root $\lambda = 1$ in the equation $\pi \cdot Q = \lambda \cdot \pi$ is one.

[2] We assume that the last vector is linked to the first one.

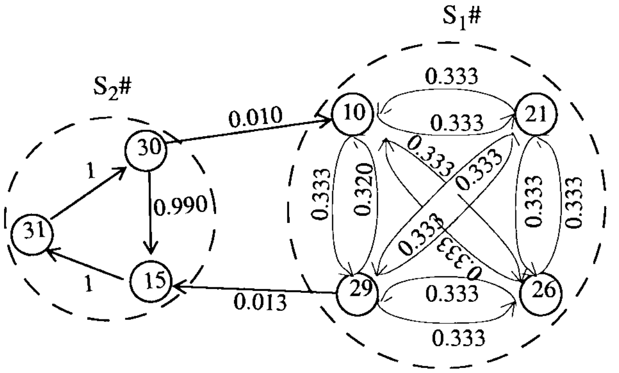


Fig. 5. The transition graph for the composite sequence S^*.

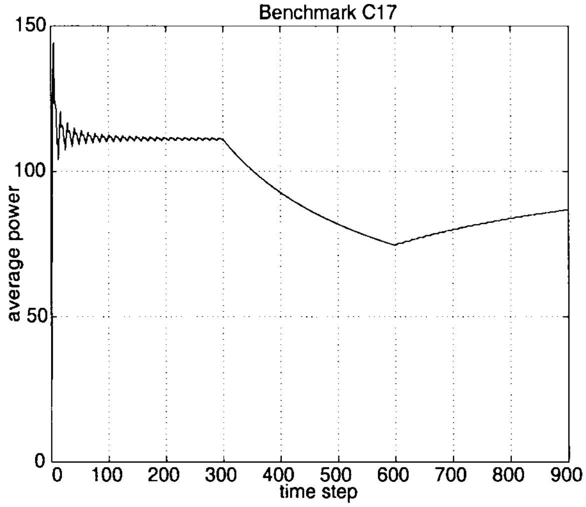


Fig. 6. Average power dissipation for C17.

$S_1^{\#}$ and $S_2^{\#}$, respectively.[3] Based on $S_1^{\#}$ and $S_2^{\#}$, we construct now a new sequence S^* which is formed by concatenating $S_1^{\#}$ and $S_2^{\#}$ for an infinite number of times [that is, $S^* = (S_1^{\#} @ S_2^{\#})^*$]; the transition graph representation of this new "macrosequence" is given in Fig. 5. The question now becomes, how will the average power consumption look like, as a function of time, when S^* is applied to any circuit? Obviously, the sequence S^* has two very different modes: one where much activity is generated at the primary inputs, and a second one where about one single input bit toggles at every time step. In Fig. 6 we can see the effect of these two different regimes on average power consumption for benchmark C17. Starting initially with $S_1^{\#}$, after 300 time steps the value of average power stabilizes around 110 μW; then, when the characteristics of the input sequence change, the power value goes down toward 70 μW and finally, due to the increase of the switching activity at the primary inputs, it comes up toward 90 μW.

This type of behavior is very common in practice. More precisely, only homogenous input sequences (which contain statistically similar vectors) will exercise the circuit such that the value of average power will converge rapidly. A typical example is a set of pseudorandom vectors where the average power value stabilizes after applying only tens of vectors. However, in practical applications, the set of stimuli may contain a mixture of vectors, each one very different as far as the average switching activity per bit is concerned.

A compaction procedure based on random walks in graphs where some pairs of vectors have very small transition probabilities, has the potential drawback of "hanging" into a small subset of states. This causes an erroneous power value, depending on which component (low or high activity) is visited more often. The same type of phenomenon is observed for statistical methods when the distribution is very different from a normal one (e.g., bimodal distributions), or when selecting from the initial sequence only the first few hundred vectors. Thus, to compact large sequences that contain nonhomogeneous power behaviors, a technique with high adaptability is needed.

We use hierarchical Markov models to structure the input space into a *hierarchy* of *macro-* and *microstates*: at the first (high) level in the hierarchy we have a Markov chain of macrostates; at the second (low) level, each macrostate is in turn characterized by a Markov chain for all its constituent microstates. Our primary motivation for this hierarchical structure is to enable a better modeling of the different stochastic levels that are present in sequences that arise in practice. As consequence, by exploiting the first level in the hierarchical Markov model, such models will make the approach highly adaptable to the behavior of the input sequence.

After constructing the hierarchy of the input sequence, starting with some macrostate, a compaction procedure with a specified compaction ratio is applied to compact the set of microstates within that macrostate. The control then returns to the higher-level in the hierarchy and, based on the conditional probabilities that characterize the Markov chain at this level, a new macrostate is entered and the process repeats until the end of the sequence (last macrostate) is reached. By doing so, we combine the advantages offered by the hierarchical model with the flexibility of the DMC modeling technique.

2) Micro/Macrostate Modeling: Having the transition graph associated to a vector sequence, our task now is to partition this transition graph into subgraphs that correspond to different behaviors (in particular, different power consumptions). To this end, we first provide some useful definitions and results.

Definition 5—Weighted Transition Graph: A *weighted* transition graph is a directed graph where any edge from state s_i to state s_j is labeled with a conditional probability $p_{ij} = p(s_j|s_i)$ *and* a weight w_{ij}[4] associated to the transition $s_i \rightarrow s_j$.

Definition 6—Weight of a Random Walk: The *weight of a random walk* in a weighted transition graph is given by

$$W = \sum_{s_i, s_j} \pi_i \cdot p_{ij} \cdot w_{ij} \tag{4}$$

where w_{ij} is the weight associated with transition $s_i \rightarrow s_j$.

Definition 7—(ε, δ)-Property: A weighted transition graph is said to have the *(ε, δ)-property* if there exists a grouping

[3] Here # is used to symbolize that sequences S_1 and S_2 are repeated for a finite number of times.

[4] We shall see later in this section the meaning of these weights for our particular application.

$\{S_1, S_2, \cdots, S_p\}$ on the set of states $\{s_1, s_2, \cdots, s_n\}$ of the transition graph satisfying

- (ε-criterion): $\forall\ s_i \in S_k,\ s_j \in S_l$ then $p(s_i|s_j) < \varepsilon$ and $p(s_j|s_i) < \varepsilon$;
- (δ-criterion): $\forall\ S_k$, for any two states $s_i,\ s_j \in S_k$, s_i, s_j connected by an edge, $\exists\ W_k$ s.t. $|W_k - w_{ij}| < \delta$. Also, if $k < l$, W_k's are such that $W_k < W_l$. S_k's are called *macrostates* whereas $s_i \in S_k$ are called *microstates* within macrostate S_k.

The intuitive reason for the above definition is that conditional probabilities from any microstate in S_k to another microstate in $S_l\,(S_k \neq S_l)$ are negligible (ε-criterion), and all transitions among microstates belonging to the same macrostate have similar weights (δ-criterion). For instance, in Fig. 5 microstates "10," "21," "26," and "29" form the macrostate $S_1^{\#}$ (with high activity), while "15," "30," and "31" form $S_2^{\#}$ (with low activity).

A particular microstate may generally appear in more than one macrostate since not only the vector itself, but also the context in which it appears is important (as in Definition 7, the weight value for a transition determines whether the microstates belong to that particular macrostate or not). Therefore, the *grouping* of states is done such that transitions are *clustered* according to their associated weights.

From what has been defined, it becomes possible to hierarchically structure the input space. Specifically, instead of considering the input sequence as a flat sequence of vectors, it can be seen as a structured multilevel discrete stochastic process called *hierarchical Markov model* (HMM). We note that HMM generalizes the familiar Markov chain concept by making each of its macrostates a stochastic model on its own, i.e., each macrostate is a Markov model as well. For instance, the graph in Fig. 5 can be represented hierarchically as shown below, where the macrostate $S_1^{\#}$ identifies the high activity mode and $S_2^{\#}$ the low activity one.

Note: It should pointed out that, in general, the high-level Markov chain is not autonomous; that is, some conditional probabilities may be different from 1. For example, if the initial sequence is structured as: $S_1@S_2@S_3@S_2@S_1$ (having thus three modes), then in the high-level Markov chain we have $p(S_3|S_2) = p(S_1|S_2) = 0.5$ (because it is equally likely to go from S_2 to either S_3 or S_1).

The initial problem of compacting a flat input sequence becomes equivalent to that of *compacting a hierarchy* of subsequences. Since vectors in each macrostate are gathered using the same δ-criterion, the compaction is now done inside each macrostate. This avoids the "hang-up" problem mentioned in Section III-B1 because all macrostates are guaranteed to be visited as their transition probabilities "scale-up" after hierarchization. For instance, in Fig. 5, the transition probabilities between $S_1^{\#}$ and $S_2^{\#}$ are 0.013 and 0.010, respectively; in the hierarchical organization shown in Fig. 7, these probabilities become one.

We now present some useful results for HMM characterization.

Theorem 2: If the state probability of each macrostate and the state probabilities for all microstates within a macrostate are preserved, then the *state probability distribution* of the initial (flat) sequence is completely captured. □

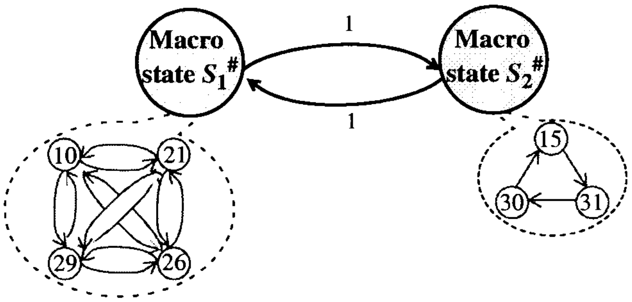


Fig. 7. A two-level hierarchy for sequence S^*.

Proof: Let s_i be any state from the original sequence. Then, its probability is given by

$$p(s_i) = \sum_{S_k} p(s_i S_k) = \sum_{S_k} p(s_i|S_k) \cdot p(S_k) = \sum_{S_k} p_k(s_i) \cdot p(S_k)$$

where $p_k(s_i)$ denotes the state probability of s_i in macrostate S_k. ■

Thus, the state probabilities are the same if they are preserved inside each macrostate and also the probability distribution for macrostates is correctly captured.

Theorem 3: Having a hierarchical model satisfying the ε-criterion, if transition probabilities of the microstates are preserved within each macrostate and if the state probabilities of macrostates are correctly captured, then the transition probabilities of the initial sequence are reproduced with an error less than or equal to ε. □

Proof: The transition probability between two states s_i and s_j can be expressed as

$$\begin{aligned} p(s_i s_j) &= \sum_{\substack{S_k \\ s_i,\, s_j \in S_k}} p(s_i s_j S_k) + \sum_{\substack{S_k,\, S_l,\, k \neq l \\ s_i \in S_k,\, s_j \in S_l}} p(s_i s_j S_k S_l) \\ &= \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) + \sum_{S_k,\, S_l} p_{kl}(s_i s_j) \cdot p(S_k S_l) \end{aligned}$$

where $p_k(s_i s_j)$ is the transition probability between microstates s_i and s_j inside macrostate S_k and $p_{kl}(s_i s_j)$ denotes the transition probability between those states if they belong to macrostates S_k, S_l, respectively. Since our assumption is that the hierarchy satisfies the ε-criterion, we have that $p_{kl}(s_i s_j) < \varepsilon$ and hence

$$\begin{aligned} p(s_i s_j) &\leq \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) + \varepsilon \cdot \sum_{S_k,\, S_l} p(S_k S_l) \\ &= \sum_{S_k} p_k(s_i s_j) \cdot p(S_k) + \varepsilon. \end{aligned}$$ ■

Thus, if the macrostate probability distribution and the transition probabilities inside each macrostate are preserved, then the actual transition probabilities are preserved up to some error ε.

Theorem 4: If the hierarchy satisfies the (ε, δ)-property (as in Definition 7), then the weight of a random walk in the flat model satisfies

$$W \le \sum_{s_k} p(S_k)\cdot W_k + \varepsilon \cdot \sum_{S_k,\, S_l,\, k\neq l} \; \sum_{\substack{s_i,\, s_j\\ s_i\in S_k,\, s_j\in S_l}} p(S_kS_l) \cdot w_{ij} + \delta \tag{5}$$

where $p(S_k)$ is the probability of being in macrostate S_k, W_k is the weight associated to macrostate S_k (as in Definition 6) and $p(S_kS_l) = p(S_k)p(S_l|S_k)$ is the transition probability from macrostate S_k to macrostate S_l. □

Proof: Let W be the weight of the initial sequence. Then, using Definition 6 and Theorem 3, we have

$$W \le \sum_{s_i,\, s_j} \sum_{S_k} p_k(s_is_j)\cdot p(S_k)\cdot w_{ij} + \varepsilon \cdot \sum_{\substack{s_i,\, s_j\\ s_i\in S_k,\, s_j\in S_l,\, k\neq l}} p(S_kS_l)\cdot w_{ij}.$$

On the other hand, if s_i, s_j are in the same macrostate S_k and the hierarchy satisfies the δ-criterion, then there is some W_k such that $|w_{ij} - W_k| < \delta$. Hence, we also have

$$W \le \sum_{s_i,\, s_j} \sum_{S_k} p_k(s_is_j)\cdot p(S_k)\cdot (W_k + \delta) + \varepsilon \cdot \sum_{\substack{s_i,\, s_j\\ s_i\in S_k,\, s_j\in S_l,\, k\neq l}} p(S_kS_l)\cdot w_{ij}$$

from where it follows the above claim. ■

In other words, a random walk on the hierarchical Markov model *preserves* up to some error the average weight of the original sequence. The first term in the above sum represents the average weight per macrostate, whereas the second accounts for the weight of transitions between them.

This general formulation applies immediately to our problem defined in Section III-A. In fact, if the input sequence is hierarchically structured, Theorem 3 guarantees that inequality (3) is still satisfied. Moreover, Theorem 4 guarantees that the average power value is maintained. This is very important from a practical point of view because the hierarchical model has the advantage of being *highly adaptive* as opposed to a flat processing of the input sequence which does well only "on average."

3) A Hamming Distance-Based Criterion for Microstate Grouping: In practice, it is hard to determine the weight w_{ij} for each individual transition. Specifically, an exact procedure would require detailed information about the circuit (e.g., its internal structure and capacitive loads) and a fast simulation procedure to derive the exact power consumption for each pair of vectors in the original sequence. In practice, such an attempt may be unsatisfactory due to the simulative overhead and the requirement to have detailed circuit information. Therefore, we adopt a different, *circuit-independent* criterion to structure the input space.

As suggested in Example 2, what we need is an indicator of the level of activity at the *primary inputs* of the circuit. To this end, we must correctly identify the different stochastic levels (subsequences) in the initial input sequence and then apply the (ε, δ)-grouping of microstates based on their associated weights. Once we identify these different subsequences in the initial sequence S^*, our compaction procedure works fine regardless of the power consumption values that would arise from the application of these two subsequences to the target circuit.

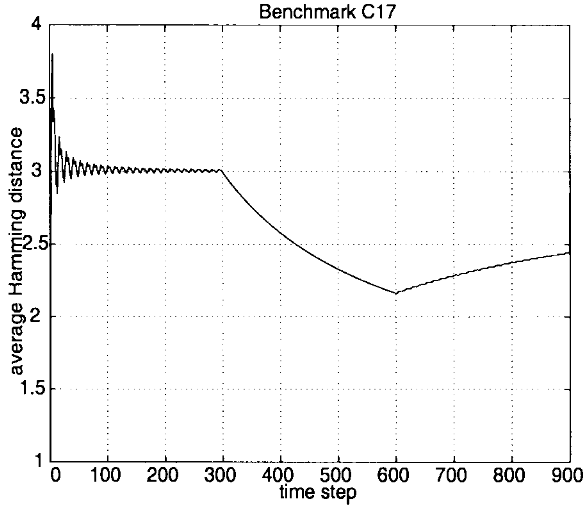


Fig. 8. Average Hamming distance variation.

To structure the input space, we propose to use the *average Hamming distance over a variable-size sliding window* because, from our investigations, it is a reliable indicator of the level of activity. We will explain subsequently how this window is actually used to calculate the average Hamming distance.

In the particular situation described in Example 2 (see Fig. 8), based on the Hamming distance criterion, the input sequence can be roughly classified into "high activity" and "low activity" macrostates.[5] While this kind of partitioning into high and low activity modes can always be used, in practice it is better to have a more refined model. For instance, if the set of all possible values for the Hamming distance is divided in three equally-sized regions that correspond to low, medium and high activity, then we can identify more than two modes of operation. A more refined model might be required in some applications where a large number of operational modes exist (e.g., an initialization mode, an active mode, a standby mode, and a sleep mode). We also note that the average Hamming distance may not always capture the specific behavior of different groups of bits in the input sequence. For instance, a criterion based on average Hamming distance may not distinguish between the two cases where either the most significant bits (MSB) are more active than the least significant bits (LSB) or vice versa. These two cases, however, may induce different power modes (even if the average Hamming distance is the same). To handle this situation, we refine the high-level Markov model as follows. Instead of having only low and high activity macrostates, we define four macrostates

- low activity LSB and low activity MSB;

[5] That is, if more than three out of five bits change, we are in the high activity mode; otherwise in the low activity one.

- low activity LSB and high activity MSB;
- high activity LSB and low activity MSB;
- high activity LSB and high activity MSB;

and, in this way, we structure the input space to achieve a stronger correlation between the behavior of the input sequence and the power modes induced by this sequence in the target circuit. In the current implementation, the user has the freedom to specify the number of groups of bits to be considered, as well as the number of macrostates per each group. Thus, at the expense of a more complex model, we are able to find a better hierarchization of the input sequence which closely follows the power modes of the target circuit.

We note that, in general, characterizing different groups of bits as far as their average Hamming distance is concerned, does not imply that we have to always consider MSB versus LSB. Instead, given the functionality/structure of the circuit and/or some knowledge about the primary input bits, we can decide what different groups of bits have to (or may be) characterized separately. We should also note, however, the advantage of the original criterion (i.e., Hamming distance), besides the low complexity, is that it only requires the availability of a meaningful input trace.

To detect the changes in the input sequence, a *variable-size sliding window* is used to compute the average value of the predictor function (in our case, the Hamming distance). At every time step, the average Hamming distance for all vectors starting with the first in the current macrostate and ending with the current vector is computed. Next, we decide whether the behavior of the sequence has changed (that is, we are in the same macrostate or not) by comparing the average Hamming distances at steps $p \cdot k$[6] and $(p+1) \cdot k$, where k is a fixed parameter (window increment size) and p is an integer. If the difference between these average Hamming distances is larger than the parameter δ set by the user, then we start with a new macrostate; otherwise, we remain within the current macrostate.

We note that the *size* of the chosen window increment should not be too small (due to the fragmentation, the high-level Markov chain becomes similar to the flat model) or too large (the low-level Markov chain becomes similar to the flat model). Our experience shows that a window increment size k of 50–100 vectors works very well in practice. We note that the ε-criterion (if satisfied) is already accounted for by this procedure since all macrostates are guaranteed to be visited (due to the scaling of conditional probabilities in the high-level model). This does not result in an incorrect value for the total power consumption, as is the case for the flat model.

C. Practical Considerations and Experimental Results

In the following, we describe a practical procedure for constructing the tree DMT_1 [15] and generating the compacted sequence. First, based on average Hamming distance criterion explained in Section III-B3, the vectors of the original sequence are assigned to macrostates. During the second traversal of the original sequence [when we extract the bit-level statistics of each individual vector and also those statistics that correspond to pairs of consecutive vectors $(v_1 v_2)$, $(v_2 v_3)$, $\cdots$, $(v_{n-2} v_{n-1})$, $(v_{n-1} v_n)$], we grow simultaneously the trees DMT_1 inside each macrostate (the low-level of the hierarchy) and also the DMT_1 tree for the sequence of macrostates (the high-level of the hierarchy). Vectors within each macrostate are sequenced together in the same DMT_1. If the input sequence satisfies the (ε, δ)-property, the transitions introduced this way do not significantly change the characteristics (average weight and transition probabilities) of the model. We continue to grow the trees at both levels of hierarchy as long as the Markov model is smaller than a user-specified threshold; otherwise we just generate the new sequence up to that point and discard (flush) the model. A new Markov model is started again and the process is continued until the original sequence is completely processed.

For the generation phase itself, we use a modified version of the dynamic weighted selection algorithm described in [22]. In general, by alternating the generation and flush phases in the DMC procedure, the complexity of the model can be effectively handled. The only remaining issue is to determine how many vectors must be generated inside each macrostate before a transition to another macrostate is performed. If a subsequence of length L_i is assigned to the macrostate S_i, after compaction with ratio r, it has to be reduced to L_i/r. We note that inside all macrostates the *same compaction ratio* should be used, otherwise the composition of the sequence (as far as power consumption is concerned) may be totally different than the composition of the initial sequence. On average, each macrostate S_i should be visited $(p(S_i) \cdot M)$ times where M is the length of the "macrosequence" (i.e., the length of the initial sequence of macrostates). Thus, each time a macrostate S_i is visited a number of $L_i/(r \cdot p(S_i) \cdot M)$ vectors needs to be generated. Since compaction is done only at the microstate level, the length of the macrosequence is preserved (the generation procedure stops when exactly M macrostates are obtained). It is also noted that this strategy does *not* allow "forbidden" vectors which means that those combinations that did not occur in the original sequence, will not appear in the final compacted sequence either.

The overall strategy is depicted in Fig. 9. Starting with an input sequence of length L_0, we perform a one-pass traversal of the original sequence to assign microstates to macrostates. Next, we simultaneously build the trees DMT_1 for the entire hierarchy (macro- and microstates). During this process, the frequency counts on DMT_1's edges are dynamically updated.

The next step in Fig. 9 does the actual generation of the output sequence (of length L). Our compaction procedure works, in principle, with two different compaction strategies. The first one is to monitor the current values of the transition probabilities and compare them with the transition probabilities of the original sequence. When the difference between the two sets of probabilities becomes sufficiently small, the generation procedure is halted. This way, we are able to satisfy any user specified error level for the transition probabilities. The second strategy is to set the compaction ratio upfront, perform compaction, and then compute the error induced by compaction *a posteriori*. In this second scenario, the user may define the largest value of the compaction ratio

[6] $p \cdot k$ represents the size of the window.

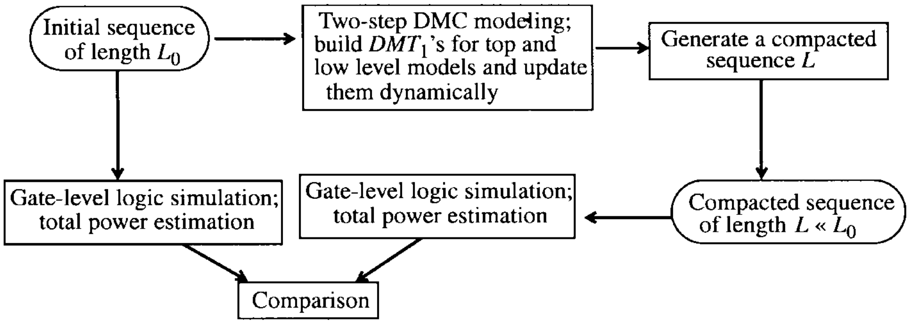

Initial sequence of length L_0
Two-step DMC modeling; build DMT_1's for top and low level models and update them dynamically
Generate a compacted sequence L
Gate-level logic simulation; total power estimation
Gate-level logic simulation; total power estimation
Compacted sequence of length $L \ll L_0$
Comparison

Fig. 9. The experimental setup for combinational circuits.

TABLE I
TOTAL POWER CONSUMPTION (μW @ 20 MHz) FOR ISCAS'85 CIRCUITS (4000 VECTORS)

			Flat model			Hierarchical model		
Circ.t	Number of inputs	Power for initial seq.	Estimated power for $r = 2$	Estimated power for $r = 5$	Estimated power for $r = 10$	Estimated power for $r = 2$	Estimated power for $r = 5$	Estimated power for $r = 10$
C432	36	1810.02	1473.30	1491.58	1230.77	1875.22	1888.42	1906.84
C499	41	4390.79	3931.30	5341.74	6126.58	4477.09	4497.01	4591.46
C880	60	3788.22	4337.18	4504.40	2803.14	3921.61	3851.42	4006.19
C1355	41	3783.35	4300.08	3065.71	4333.81	3828.45	3910.45	3933.25
C1908	33	6352.07	4947.08	4565.87	7094.39	6127.82	6145.49	6493.44
C3540	50	14471.46	17153.07	9005.19	3527.65	14797.30	15056.43	15021.08
C6288	32	104158.25	86586.18	81100.59	82652.47	101407.12	98112.01	96295.36
		Avg.% err.	16.40	23.64	31.45	2.67	3.55	4.74

based on the stationarity hypothesis which should be satisfied on any segment of the original sequence that is compacted. More precisely, the shortest subsequence where the stationarity hypothesis must be satisfied limits the highest compaction ratio that can be achieved. No matter what strategy is used, if the initial sequence has the length L_0 and the newly generated sequence has the length $L < L_0$, then the outcome of this process is a compacted sequence, equivalent to the initial one as far as total power consumption is concerned; we say that a *compaction ratio* of $r = L_0/L$ was achieved.

Finally, a validation step is included in the strategy; we resorted to an in-house gate-level logic simulator developed under SIS.[7] The total power consumption of some ISCAS'85 benchmarks has been measured for the initial and the compacted sequences, making it possible to assess the effectiveness of the compaction procedure (under both zero- and real-delay models).

In Table I, we provide the real-delay results for a set of highly biased sequences (of length 4000) which represent input stimuli for real applications and have been provided to us by a chip manufacturer. We point out that while we had access to these real input sequences, we could not obtain also the description of the actual chips. Consequently, we used these real input sequences with standard benchmarks taken from ISCAS suite and evaluated the quality of the results. In the end, we were pleased to learn that the error we found experimenting on standard benchmarks was consistent with the error found independently by the manufacturers. This consistency is not surprising because, at least if our assumptions are satisfied, our approach is essentially independent of the target circuit.

The sequences in Table I contain three types of subsequences: a high activity subsequence, followed by a low activity subsequence, and finally by a pseudorandom one. To get a deeper intuition about the characteristics of these sequences, we will consider subsequently the particular case of testbench C6288 which will be also referred later in this section. The bit-level switching activities (for the input sequence of C6288) vary in the range of 0.24 to 0.53 for the first subsequence (vectors 1–2500), 0.17 to 0.61 for the second one (vectors 2501–3700), and finally, 0.45 to 0.54 for the pseudorandom sequence (vectors 3701–4000). In terms of the average Hamming distance variation, the values are 10.33, 9.91, and 16.11 for the first, second, and third subsequence, respectively. We note that these average numbers do not fully illustrate the differences in characteristics among these three subsequences and, therefore, we present in Fig. 10 the variation of the Hamming distances associated to these sequences.

From this graph, we can see that the Hamming distance varies much faster for the first subsequence compared to the second one; this abrupt change in the characteristic of the input sequence can be noticed after the first 2500 vectors. On the other hand, after 3700 vectors, we enter in the pseudorandom domain where the Hamming distance stabilizes around value 16. These three very different subsequences induce different

[7] For more accuracy, a switch-level simulator (e.g., Power Mill [3]) may be used.

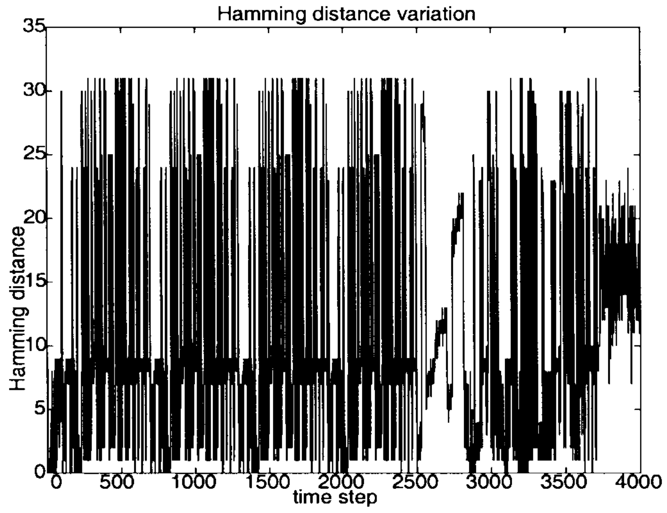


Fig. 10. Hamming distance variation for C6288.

power consumptions in the circuit C6288. For instance, the power dissipated during the first mode is 96 375.08 μW, 68 231.38 μW for the second one, and finally, 338 948.28 μW for the pseudorandom subsequence. While the numbers are slightly different for the other circuits reported in Table I, the qualitative explanation and the details given here for C6288 remain valid for all other examples.

As shown in Table I, the initial sequences were compacted with three different compaction ratios (namely $r = 2$, 5, and 10). This table includes the total power dissipation measured for the initial sequence (column 3) and for the compacted sequence (columns 4–9). The time in seconds (on a Sparc 20 workstation with 64 MB of memory) necessary to read and compress data with DMC modeling was below 5 s in all cases, either for the flat or for the hierarchical model. Since the compaction with DMC modeling is linear in the number of nodes in the structure DMT_1, this value is far less than the actual time needed to simulate the uncompacted sequence. During these experiments, the number of states allowed in the Markov model was 20 000, on average (about 500 KB). These sequences satisfied the ε-criterion for $\varepsilon = 0.001$, while the parameter δ in Definition 7, was set to be 0.05 · (# of input bits). (This corresponds to having up to 20 macrostates in the hierarchical model.)

As we can see in Table I, the quality of results is very good, even when the length of the initial sequence is reduced by one order of magnitude. Thus, for C432 in Table I, instead of simulating 4000 vectors with an exact power of 1810.02 μW, only 800 vectors with an estimate of 1888.42 μW or just 400 vectors with a power consumption estimated as 1906.84 μW may be used. This reduction in the sequence length has a significant impact on speeding-up the simulative approaches where the running time is proportional to the length of the sequence which must be simulated. For instance, using the PowerMill simulator [3], the average speed-up obtained for simulation time of benchmarks and sequences in Table I is 11. By comparison, if the flat model is used for the same benchmark, the relative errors in power prediction are 18% and 32%, respectively. The primary reason for this inaccuracy is the lack of adaptability which characterizes the flat model when applied to multimodal sequences.

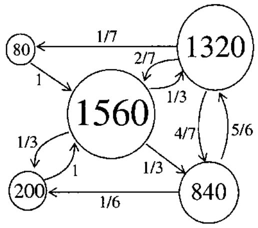


Fig. 11. The two-level hierarchy for C6288.

TABLE II
RESULTS OBTAINED FOR SIMPLE RANDOM SAMPLING

	Number of vector pairs		Error violations (%)		
Circuit	Max.	Avg.	> 5%	> 6%	>10%
C432	3300	2176	1.1	0.7	0.4
C499	1500	862	1.4	1.3	0.2
C880	3990	2705	1.8	0.4	0.7
C1355	1380	814	1.7	1.0	0.2
C1908	1620	846	1.9	1.3	0.2
C3540	2340	1446	2.0	1.3	0.4
C6288	7470	5422	1.4	1.4	0.3

Once again, we note that the assumption of stationarity is essential for our compaction procedure. Moreover, even if the stationarity hypothesis is satisfied, the convergence result in Corollary 2 states that if the error on input conditional probabilities ε converges to zero, then the difference between the total power values converges also to zero. However, we were not able to prove that this convergence is *monotonic,* so oscillations may occur for a specific run. It is however expected that the general tendency is to get a smaller error for a smaller compaction ratio when a large number of runs is taken into consideration.

We present in Fig. 11 a typical case, that is the two-level hierarchy obtained for the benchmark C6288 in Table I. We indicate for each macrostate the number of microstates included in it; for instance, the largest macrostate contains 1560 microstates, while the smallest has only 80 microstates. As we can see, for these particular values of parameters ε and δ, we have identified five modes (macrostates) in the input sequence, each one containing a very different number of microstates. We also note that the conditional probabilities at the highest level of the hierarchy scaled-up after hierarchization.

Finally, we compare our results generated by HMM with simple random sampling of vector pairs from the original sequences. In simple random sampling, we performed 1000 simulation runs with 0.99 confidence level and 5% error level on each circuit.[8] We report in Table II the maximum and average number of vector pairs needed for total power values to converge [1], [4]. We also indicate the percentage of error violations for total power values, using as thresholds 5%,

[8] This means that the probability of having a relative error larger than 5% is only 1%.

TABLE III
RESULTS OBTAINED FOR HMM APPROACH

		Error violations (%)		
Circuit	Number of vectors	> 5%	> 6%	>10%
C432	800	2.6	0.6	0.0
C499	800	0.1	0.0	0.0
C880	800	2.8	0.9	0.0
C1355	800	0.1	0.0	0.0
C1908	800	0.1	0.0	0.0
C3540	1200	1.6	0.3	0.0
C6288	3600	1.5	0.0	0.0

6%, and 10%. Using different seeds for the random number generator (and, therefore, choosing different initial states in the sequence generation phase), we run a set of 1000 experiments for the HMM technique. In Table III, we give the results obtained with the hierarchical model, for the same thresholds used in simple random sampling.

Once again, the results obtained with HMM modeling technique score very well and prove the robustness of the present approach. As we can see, using fewer vectors, the accuracy of HMM is higher than the one of simple random sampling in most of the cases. Noteworthy examples are benchmarks C499, C1908, and C6288, where less than 60% of the maximal number of vector pairs needed in random sampling for convergence are sufficient for HMM to achieve higher accuracy and confidence levels.

In Table IV, we provide the real-delay results for a set of long sequences having the length of 200 000 vectors. We generated these sequences by setting the most significant half of the bits to realize a counted sequence and the remaining bits to be random. For the typical example C6288, we had two different modes in the input sequence, a low- (vectors 1–100 000) and a high-activity one (vectors 100 001–200 000). The bit-level switching activity varies in the range of 0–0.67 for high-activity subsequence and is 0.5 for the low-activity one. The average Hamming distance is 8.48 and 16 for the first and second subsequence, respectively. In terms of power dissipation, for the first and second regimes, the values are 593.75 μW and 246 763.92 μW, respectively.

The sequences in Table IV were compacted with three different compaction ratios (namely, 10, 100, and 1000) and the total power consumption was estimated for each individual benchmark. Using a Sparc 20 workstation with 64 MB of memory, the time necessary to read and compress data was less than 10 s in all cases.

As we can see, the quality of results is very good, even when the length of the initial sequence is reduced by two or three orders of magnitude. This reduction in the sequence length has a significant impact on speeding-up the simulative approaches where the running time is proportional to the length of the sequence which must be simulated.

IV. SEQUENCE COMPACTION FOR SEQUENTIAL CIRCUITS

The approach presented in Section III is applicable only to combinational circuits because it considers only first-order temporal effects (i.e., pairs of consecutive vectors) to perform sequence compaction. In the case of FSM's, this is not sufficient. In this section, we present a solution for compacting an initial sequence such that the steady-state and transition probabilities of the signal lines are almost the same.

A. Finite-Order Memory Effects

Information about the steady-state and transition probabilities is very important because both of them completely characterize the FSM behavior. In particular, they have immediate application in power estimation area because, for sequential circuits, temporal correlations longer than one time step can affect the overall behavior of the FSM and, therefore, result in very different power consumptions. This point is illustrated by a simple example.

Example 3: Let S_1 and S_2 be two 4-bit sequences of length 26, as shown in Fig. 12(a). These two sequences have exactly the same set of first-order temporal statistics as shown in Fig. 12(b). In this figure, we provide the wordwise transition graph for these two sequences where the topmost bit is the most significant bit (e.g., in S_1, $v_1 = v_2 =$ "1," $v_3 =$ "2," $\cdots$, $v_{26} =$ "9"). Suppose now that S_1 and S_2 are fed to the benchmark $s8$ from the $mcnc$'91 sequential circuits suite. Looking at different internal nodes of the circuit, it is noted that the total number of transitions made by each node is very different when the circuit is simulated with S_1 or S_2. Moreover, the total power consumption at 20 MHz is 384 and 476 μW, respectively, showing a difference of more than 24% even for this small circuit. This raises the natural question, "why does this difference appear?," despite the fact that S_1 and S_2 have the same STG shown in Fig. 12(b). The reason for this different power dissipation is that S_1 and S_2 have a different set of second-order statistics; that is, the sets of triplets (three consecutive vectors) are quite different. For instance, the triplet (1, 2, 7) in S_2 does not occur in S_1; the same observation applies to the triplet (5, 2, 3) in S_2. The conclusion is that having the same set of one-step transition probabilities *does not* imply that the set of second-order or higher-order statistics are identical and, as it was just illustrated in this small example, higher-order statistics can make a significant difference in FSM's total power consumption. The initial problem of compacting an initial input sequence can be now cast in terms of power as follows: can we transform a given input sequence into a shorter one, such that the new body of data is a good approximation of the initial sequence as far as total power consumption is concerned?

B. Problem Formulation

The focus is now turned from the input sequence to the actual target circuit and to the investigation of the effect of input statistics on its transition probabilities (primary inputs and present state lines). As shown in Fig. 13, we model the "tuple" (*input_sequence*, *target_circuit*) by the "tuple" (*Markov_chain*, *FSM*), where *Markov_chain* models the *input_sequence* and *FSM* is the sequential machine where the transition probabilities have to be determined. In what follows, x_n, s_n will denote the random variables associated to the

TABLE IV
TOTAL POWER (μW @ 20 MHz) FOR LONG SEQUENCES (200 000 VECTORS)

Circuit	Number of inputs	Power for initial seq.	Estimated power for $r = 10$	Estimated power for $r = 100$	Estimated power for $r = 1000$
C432	36	2817.36	2694.91	2990.50	3033.51
C499	41	5654.54	5488.53	5479.64	5425.83
C880	60	6173.67	6153.70	6241.48	5762.76
C1355	41	4884.23	4754.19	4755.69	4671.19
C1908	33	7524.10	7413.19	7376.26	7354.60
C3540	50	19250.26	18158.39	19649.61	20014.07
C6288	32	123678.29	111104.11	113955.95	113658.88
		Avg. % err.	2.64	2.99	5.07

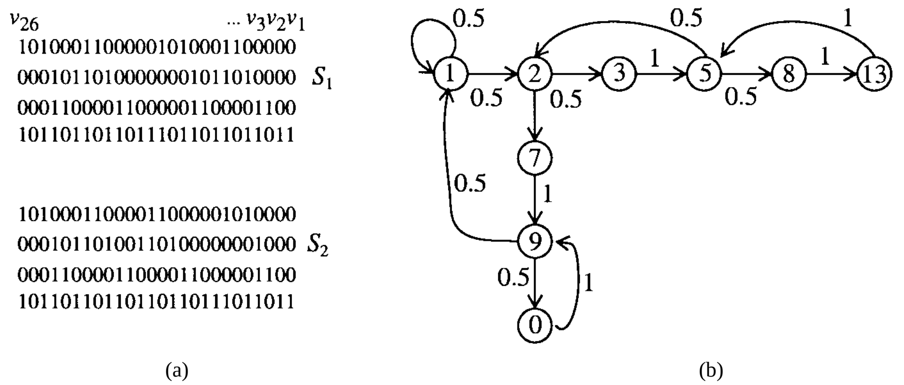


Fig. 12. Two sequences with the same first-order statistics.

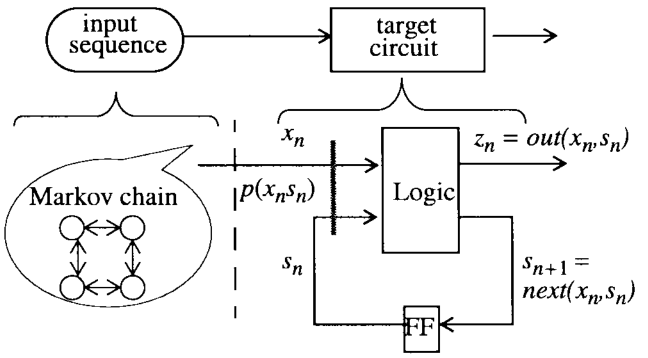


Fig. 13. The tuple (*Markov-Chain, FSM*).

inputs and state lines of the target sequential machine; $p(x_n s_n)$ is the probability that, at time step n, the input is x_n and the state is s_n.

We are interested in defining the joint probabilities $p(x_n s_n)$ and $p(x_n s_n x_{n-1} s_{n-1})$ because, as shown in Fig. 13, they completely capture the characteristics of the input (primary inputs *and* present state lines) that feeds the next state and the output logic of the target circuit. In addition, having defined the joint random variables "$x_n s_n$" and "$x_n s_n x_{n-1} s_{n-1}$," the vector compaction problem for sequential circuits becomes essentially the vector compaction problem for combinational circuits as presented in Section III-A.

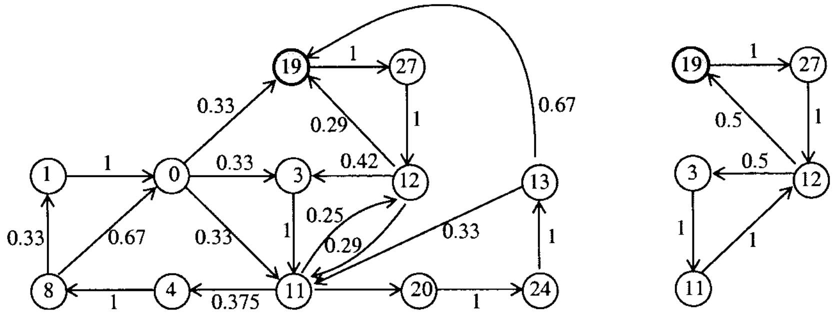
Problem Formulation: For a sequence of length L_0, find another sequence of length $L < L_0$ (consisting of a subset of the initial sequence), such that the *average joint transition probability* on the primary inputs *and* present state lines is preserved *wordwise,* for *two* consecutive time steps. More formally, the following condition should hold:

$$|p(x_n s_n x_{n-1} s_{n-1}) - p^*(x_n s_n x_{n-1} s_{n-1})| < \varepsilon \tag{6}$$

where p and p^* are the probabilities in the original and compacted sequences, respectively. □

This condition simply requires that the joint transition probability for inputs and states $(x_i s_i)$ is preserved within a given level of error for two consecutive time steps. We note that, because the vector compaction problem is given in terms of joint probabilities $p(x_n s_n)$ and $p(x_n s_n x_{n-1} s_{n-1})$, the proof of correctness in Section III-A remains also valid for the sequential case. Once again, the stationarity condition on primary inputs *and* state lines of the FSM is essential for this result.

Finally, we point out that in our approach, we do not make any attempt to compact the subsequence in the original sequence that is used for the initialization of the target FSM. More precisely, if the original sequence S consists of two components, one being the initialization sequence of the target FSM (S_{init}) (usually 50–100 vectors) and the other being a regular set of vectors that excite the FSM (S_{reg}), then $S = S_{init}@S_{reg}$. We do consider for compaction only the S_{reg} component. (Usually, $S_{init} \ll S_{reg}$ and thus S_{reg} is the prime candidate for compaction.) This is primarily because we do not want to alter the initializability properties of the FSM and also because we need stationary behavior on the state lines of the FSM to have our compaction procedure work.



(a) (b)

Fig. 14. Two STG's obtained for the signal lines $(x_n s_n)$ of benchmark $dk17$.

C. The Effect of Finite-Order Statistics on FSM Behavior

We will investigate now the conditions that can guarantee that relation (6) is satisfied. Under the general assumptions of *stationarity* and *ergodicity* [18], the following result can be proved.

Theorem 5: If the sequence feeding a target sequential circuit has order k, then a lag-k Markov chain which correctly models the input sequence, also correctly models the k-step conditional probabilities of the primary inputs *and* internal states, that is

$$p(x_n s_n | x_{n-1}s_{n-1}x_{n-2}s_{n-2} \cdots x_{n-k}s_{n-k}) = p(x_n|x_{n-1}x_{n-2} \cdots x_{n-k}). \quad (7)$$

Proof: Since x_n is a lag-k Markov chain, we can first prove that, for any $n \geq k+1$, the following holds:

$$\begin{aligned} p(x_n s_{n-k} | x_{n-1}x_{n-2} \cdots x_{n-k}) &= p(x_n|x_{n-1}x_{n-2} \cdots x_{n-k}) \\ &\quad \cdot p(s_{n-k}|x_{n-1}x_{n-2} \cdots x_{n-k}). \end{aligned}$$

Let $p(x_n s_n x_{n-1}s_{n-1} \cdots x_{n-k}s_{n-k})$ be the joint transition probability for inputs and states over k consecutive time steps. Then we have

$$p(x_n s_n \cdots x_{n-k}s_{n-k}) = \begin{cases} p(x_n x_{n-1} \cdots x_{n-k}s_{n-k}), & \text{if } next(x_i, s_i) = s_{i+1} \\ & \quad i = n-k, \cdots, n-1 \\ 0, & \text{otherwise.} \end{cases}$$

For the first alternative we have

$$\begin{aligned} p(x_n s_n \cdots x_{n-k}s_{n-k}) &= p(x_n x_{n-1} \cdots x_{n-k}s_{n-k}) \\ &= p(x_n s_{n-k}|x_{n-1} \cdots x_{n-k}) \cdot p(x_{n-1} \cdots x_{n-k}) \end{aligned}$$

which becomes

$$\begin{aligned} p(x_n s_n \cdots x_{n-k}s_{n-k}) = &\, p(x_n|x_{n-1}x_{n-2} \cdots x_{n-k}) \\ &\cdot p(x_{n-1}x_{n-2} \cdots x_{n-k}s_{n-k}). \end{aligned}$$

Dividing both sides by $p(x_{n-1}x_{n-2} \cdots x_{n-k}s_{n-k})$, and using the fact that

$$\begin{aligned} p(x_n s_n \cdots x_{n-k}s_{n-k}) &= p(x_n x_{n-1} \cdots x_{n-k}s_{n-k}) \\ &= p(x_n s_{n-k}|x_{n-1} \cdots x_{n-k}) \cdot p(x_{n-1} \cdots x_{n-k}) \end{aligned}$$

we obtain exactly (7). For the second alternative, if $x_n s_n x_{n-1}s_{n-1}x_{n-2}s_{n-2} \cdots x_{n-k}s_{n-k}$ is not a valid sequence, then $p(x_n s_n|x_{n-1}s_{n-1}x_{n-2}s_{n-2} \cdots x_{n-k}s_{n-k}) = 0$ and this concludes our proof. ■

We note, therefore, that, preserving order-k statistics for the inputs implies also that order-k statistics will be captured for inputs and state lines. As long as the order k correctly models the input sequence, we cannot produce new transitions of the FSM and, therefore, "forbidden" subsequences of order k. However, by using DMT$_k$, we may introduce new subsequences of order higher than k but this is irrelevant for the functionality of the FSM as long as we are guaranteed by Theorem 5 that the conditional probabilities (and thus steady state probabilities) for inputs and state lines are preserved.

We also note that, in particular, the first-order statistics of the joint probabilities defined in (6) are implicitly preserved. However, modeling a k-order source with a lower order model may introduce accumulative inaccuracies. From a practical point of view, this means that underestimating a high-order source, it is possible not to correctly preserve even the first-order transition probabilities and thus violate the requirement in (6). In terms of power consumption, this will adversely affect the quality of the results as we can see from the following example.

Example 4: Once again the sequences S_1 and S_2 in Example 1 are considered and assume that they feed the benchmark $dk17$. It will be illustrated that indeed, if the input sequence has order two, then modeling it as a lag-one Markov chain will *not* preserve the first-order joint transition probabilities (primary inputs and internal states) in the target circuit. We simulated the benchmark $dk17$ (starting with the same initial state "19") for both sequences and we present in Fig. 14 [Fig. 14(a) is for S_1 and Fig. 14(b) is for S_2] the wordwise transition graphs obtained for the signal lines $(x_n s_n)$. The benchmark $dk17$ has two primary inputs and three flip-flops, therefore, in Fig. 14, any node is decimally encoded using five bits. For instance, the initial state "19" corresponds to the binary code "10 011" that is, "10" for primary inputs and "011" for state lines. Starting from state "19" and applying "11" on the primary inputs, the present state becomes "011," therefore, we enter the node "11 011" $=$ 27 in Fig. 14. As we can see, because S_1 can be modeled as a first-order Markov source, while S_2 must be modeled as a second-order Markov source, the corresponding transition graphs are quite different. From a practical point of view, this means that if one high-order source is underestimated (for instance, assuming that second- or higher-order temporal correlations are not important), then even the first-order transition probabilities in the target circuit may not be preserved. In terms of power consumption, this adversely affects the quality of the results as shown and discussed in Example 3.

Corollary 3: If the sequence feeding the target circuit has order one, then a lag-one MC will suffice to model correctly the joint transition probabilities of the primary inputs and internal states in the target circuit, that is

$$p(x_n s_n | x_{n-1} s_{n-1}) = p(x_n | x_{n-1}). \tag{8}$$

In other words, if the sequence feeding the target circuit can be accurately modeled as a first-order Markov chain, then a first-order power model can be successfully applied because the whole "history" of the input is limited to only two consecutive time steps. In this particular case, the compaction problem for both FSM's and combinational circuits becomes essentially the same; that is, all that is needed is to efficiently model the input sequence as a first-order Markov model.

D. Fixed and Variable-Order Markov Sources

1) The Order of a Markov Source: The next step is to determine the order of a Markov source, because, as proved in Theorem 5, knowing the correct value of k, is essential for FSM analysis. To this end, based on the probability of finding a "block" of vectors $\langle v_n, v_{n-1}, \cdots, v_1 \rangle$[9] in any sequence in S [denoted by $p(v_n v_{n-1} \cdots v_1)$], we introduce the following entropy-like quantities.

Definition 8—Block Entropy: The *block entropy* of length n is defined as

$$H_n = \sum_{\langle v_n v_{n-1} \cdots v_1 \rangle} p(v_n v_{n-1} \cdots v_1) \cdot \log p(v_n v_{n-1} \cdots v_1) \tag{9}$$

where $n \geq 1$.

Definition 9—Conditional Entropy: The *conditional entropy* associated with the addition of a new vector v_{n+1} to the left of an already existing block $\langle v_n, v_{n-1}, \cdots, v_1 \rangle$ is defined as

$$h_n = H_{n+1} - H_n \tag{10}$$

for $n \geq 1$ and $h_0 = H_1$.

Definition 10—Source Entropy: The *entropy of a source,* or the *uncertainty per step,* is defined as

$$h = \lim_{n \to \infty} \frac{H_n}{n} = \lim_{n \to \infty} h_n \tag{11}$$

and often referred to as *metric entropy* or *entropy rate* [19].

For stationary and ergodic processes, Khinchin [23] has shown that H_n in (9) is monotonically increasing, H_n/n is monotonically decreasing and the limit in (11) exists. We can justify the term "conditional" used for the entropy in (10) by proving that

$$\begin{aligned} h_n =& H_{n+1} - H_n \\ =& \sum_{\langle v_{n+1} v_n \cdots v_1 \rangle} p(v_{n+1} | v_n v_{n-1} \cdots v_1) \cdot p(v_n v_{n-1} \cdots v_1) \\ & \cdot \log p(v_{n+1} | v_n v_{n-1} \cdots v_1) \end{aligned} \tag{12}$$

which illustrates the mean uncertainty of a new vector v_{n+1}, given the knowledge of vectors $v_n, v_{n-1}, v_{n-2}, \cdots, v_1$. Therefore, the conditional entropy h_n is a measure of the predictability for the whole process.

Lemma 1: For any lag-k Markov chain, it holds that $H_n = H_k + (n-k) \cdot (H_{k+1} - H_k), \forall n \geq k$. □

Proof: We first show that $h_n = h_k \; \forall n \geq k$. Then, we can also write

$$\begin{aligned} h_k &= H_{k+1} - H_k \\ h_{k+1} &= H_{k+2} - H_{k+1} \\ &\vdots \\ h_{k+n-1} &= H_n - H_{n-1}. \end{aligned}$$

From here by summation (after reducing the appropriate terms) we get: $(n-k) \cdot h_k = H_n - H_k$; using (10) we get the above claim. ■

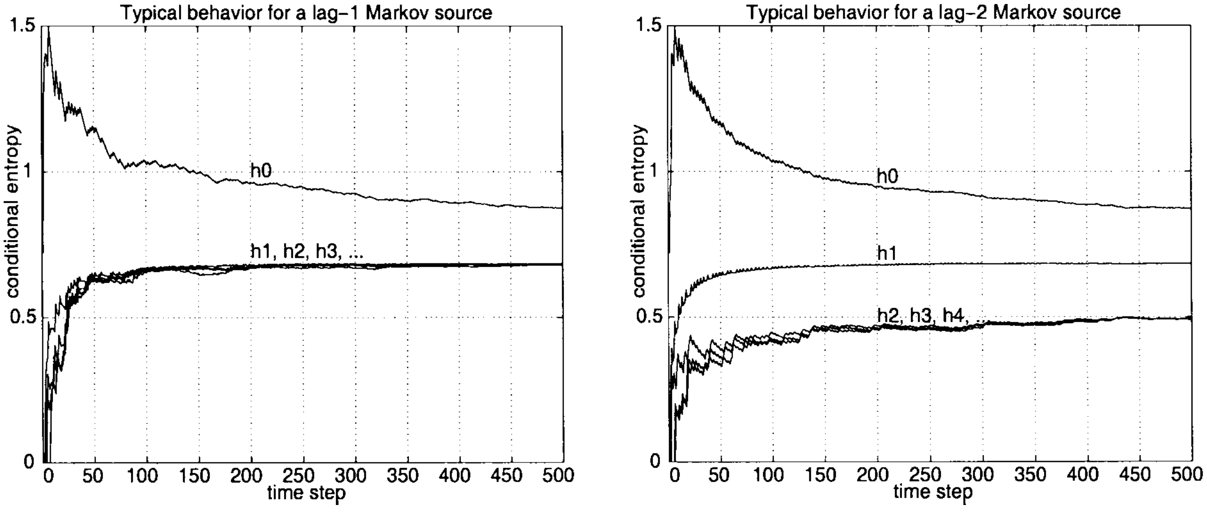
Due to statistical correlations extending over a range of only k iterations, for Markov sources of order k, the conditional uncertainty h_n is decreasing until it eventually reaches its limit value h for $n = k$ [23]. The memory effects of the source are reflected by this *saturation point* which means that value of n when the limit h is reached exactly or with a good approximation.

Example 5: In Fig. 15 we have the typical behavior for a lag-one and a lag-two Markov sequences, that have been generated by using a first– and a second-order recursive relations, respectively. The two sources have the same order-zero and order-one statistics, but different order-two statistics. In both cases, the n-step conditional entropy h_n of the source reaches its limit h after few tens of vectors. In the first case, the limit is $h = h_1$, while in the case of the lag-two Markov source, the limit is given by h_2 as expected according to the above discussion.

Finally, we note that there may exist different subsequences of order k within a given initial sequence. For instance, we may have an initial sequence S which consists of two very different second-order subsequences, S_1 and S_2, generated with a second-order Markov source (see Fig. 16). In this case, the block entropy in Definition 8 (and implicitly the source entropy in Definition 10) will be different for these subsequences (despite the fact that both of them exhibit the same type of second-order temporal correlations) and this is enough for our approach to make the distinction. In Fig. 15, we plot the variation of the conditional entropies as function of time. We observe that, indeed, after time step 500, the conditional entropies h_0, h_1 stabilize to different values for S_1 and S_2.

2) Composite Sequences: In practice, lag-k Markov chains with fixed k may not be a good enough approximation for a given sequence. For example, a real sequence can

[9] Because the process is stationary and ergodic, in this notation, the lower indices 1, 2, $\cdots$, n do not designate the absolute time instances, but the sequencing among the vectors that occur within a block of length n.



(a) (b)

Fig. 15. The behavior of conditional entropies for a lag-one and a lag-two Markov sequences.

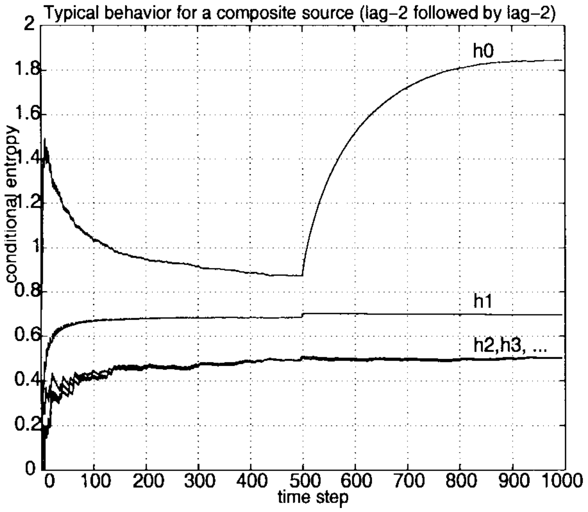


Fig. 16. The behavior of conditional entropies for two lag-two Markov subsequences.

be a mixture of sequences, each generated from a different order Markov source. We call such a sequence a *composite sequence* to emphasize its nonhomogeneous characteristics. When the sequence changes its behavior due to the change in order, the stationarity and convergence hypotheses no longer hold. For instance, if we consider two mixed sequences, the first containing a lag-one followed by a lag-two Markov subsequence, and the second a lag-two followed by a lag-one Markov subsequence, the behavior of conditional entropies is the one depicted in Fig. 17.

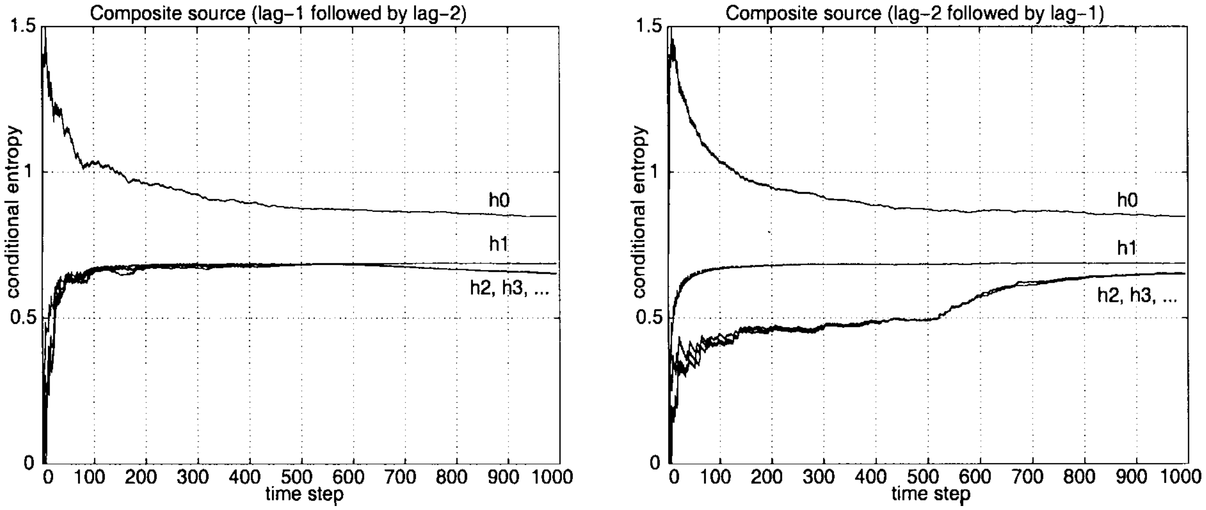
In the first case, after a sufficiently large number of steps, h_1 and h_2 do not remain the same. Actually, in the long run, the limit in (11) becomes h_2, thus detecting an order of two for the source. In this case, the convergence to h_1 is violated and thus the sequence changes its order from one to two. In the second case, after already being stabilized to the stationary values, the conditional entropies h_2, h_3, h_4, $\cdots$, tend to increase so that the order of the second half of the sequence can no longer be considered two.

This discussion provides a starting point for determining the order of subsequences in a dynamic fashion. The only requirement that has to be satisfied is the stationarity of each subsequence and, in this case, the entire sequence is called *piecewise stationary*. This is the basic hypothesis of all our subsequent experiments. To test the condition of piecewise stationarity, in practice we can use again the metric entropy in conjunction with support from the simulation of the STG of the machine. We also note that, for sequential circuits, we consider piecewise stationary sequences that exhibit only a single power mode. However, for sequences that have multiple power modes, the hierarchization procedure in Section III can be still applied but, in this case, we have to consider the average Hamming distance on the primary inputs *and* state lines of the FSM to build the hierarchical model. This implies that some knowledge about the behavior and state encoding of the FSM are available to the user.

3) Variable-Order Dynamic Markov Models: From results shown in the previous sections, we need an efficient way to model lag-k Markov chains that could characterize the input sequences that feed the target circuit. The structure DMT_1 used by authors in [15] is general enough to completely capture the correlations among all bits of the same input vector and also between successive input patterns. However, it has conceptually no inherent limitation to be further extended to capture temporal dependencies of higher orders. For instance, if we continue to define recursively DMT_2 (starting with DMT_1), we can basically capture second-order temporal correlations. For any sequence where v_i, v_j, v_k are three consecutive vectors (that is, $v_i \rightarrow v_j \rightarrow v_k$), the tree DMT_2 looks like in Fig. 18. The following result, gives the theoretical basis for using the dynamic Markov trees to capture high-order temporal correlations.

Theorem 6: The general structure DMT_k and its parameters completely capture spatial and temporal correlations of order k. □

Proof: Let $v = v_0 v_1 \cdots v_k$ be a string in DMT_k (the substring v_i belongs to the i-level tree, DMT_i). We have that



(a) (b)

Fig. 17. Typical behavior of conditional entropies for composite sequences.

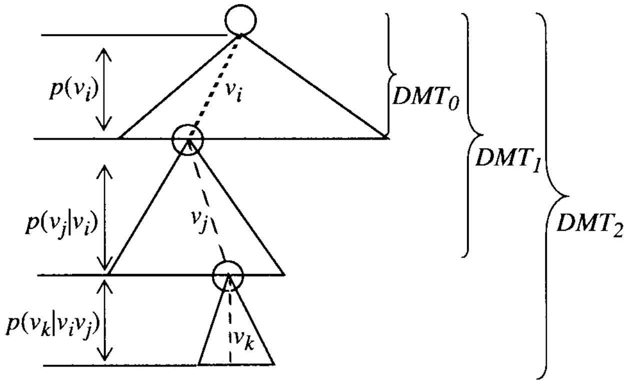


Fig. 18. A second-order dynamic Markov tree.

$p(v_k|v_{k-1}v_{k-2}\cdots v_0) = p(v_0v_1\cdots v_k)/p(v_0v_1\cdots v_{k-1})$ and thus the lag-k Markov chain characterizing the input can be fully modeled by the DMT$_k$ structure. ■

E. Practical Considerations and Experimental Results

The DMC modeling approach offers the significant advantage of being a *one-pass adaptive technique*. As a one-pass technique, there is no requirement to save the whole sequence in the on-line computer memory. Starting with an initial empty tree DMT$_K$, while the input sequence is scanned incrementally, both the set of states and the transition probabilities change dynamically making this technique highly adaptive. Also, using this data structure, we can easily account for conditional entropies and detect the order of the Markov source. Under stationarity conditions, the order is detected as the minimum k such that $|h_k - h_n| < \varepsilon$, for some $\varepsilon > 0$ and any $n = k+1, \cdots, K$ where K is the maximum order of the source to be detected. After that, if either this condition becomes violated or the stationarity hypothesis does not hold, the model is flushed and restarted. As in the combinational case, for each dynamically grown tree, the generation phase is driven by the user-specified compaction parameter ratio r. We also note that, as long as the order k models correctly the input sequence, we cannot produce new transitions of the FSM and, therefore, "forbidden" subsequences of order k. (This is guaranteed by Theorem 5.) This is an essential capability needed to avoid "hang-up" ("forbidden") states of the sequential circuit during the simulation process for power estimation.

The overall strategy is shown in Fig. 19. We assume that the input data is given in the form of a sequence of binary vectors. Starting with an input sequence of length L_0, we perform a one-pass traversal of the original sequence and simultaneously build the basic tree DMT$_K$; during this process, the frequency counts on DMT$_K$'s edges are dynamically updated. During this process, the order of the source is also determined. The generation step is done using a modified version of the weighted selection algorithm [22]. Finally, a validation step is included in the strategy; we have used an in-house gate-level logic simulator developed under SIS. The total power consumption of some sequential benchmarks has been measured for the initial and the compacted sequences, making it possible to assess the effectiveness of the compaction procedure.

Table V shows the gate-level power simulation results obtained for composite sequences of length 10 000. The hybrid character of these sequences makes a significant difference in terms of total power consumption for all benchmarks. These sequences have been generated using different generators and exhibit temporal correlation of various orders. More precisely, we have subsequences of order of two (of length 3000 vectors), followed by subsequences of order one (of length 4000) and finally, once again by subsequences of order two. As first-order generators we use counted sequences restarted at random after a fixed number of patterns have been generated. As second-order generators, we use sources of information based on Fibonacci sequences. Basically, we generate (on the appropriate number of bits) Fibonacci sequences started at random after a fixed number of patterns have been generated. (To avoid the generation of completely deterministic sequences, we also add to the pure Fibonacci generator a low-level white noise using a standard random number generator.) Because

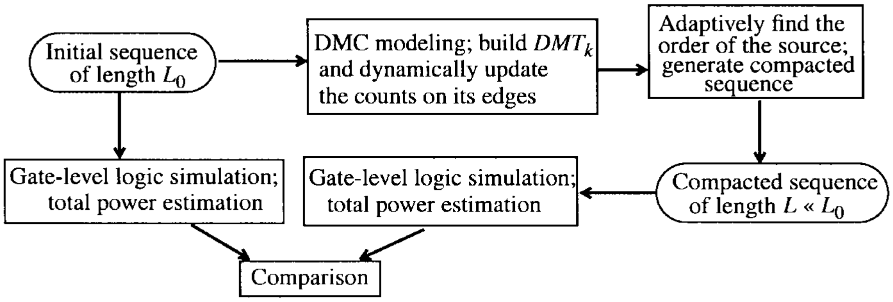

Initial sequence of length L_0
DMC modeling; build DMT_k and dynamically update the counts on its edges
Adaptively find the order of the source; generate compacted sequence
Gate-level logic simulation; total power estimation
Gate-level logic simulation; total power estimation
Compacted sequence of length $L \ll L_0$
Comparison

Fig. 19. The experimental setup for sequential circuits.

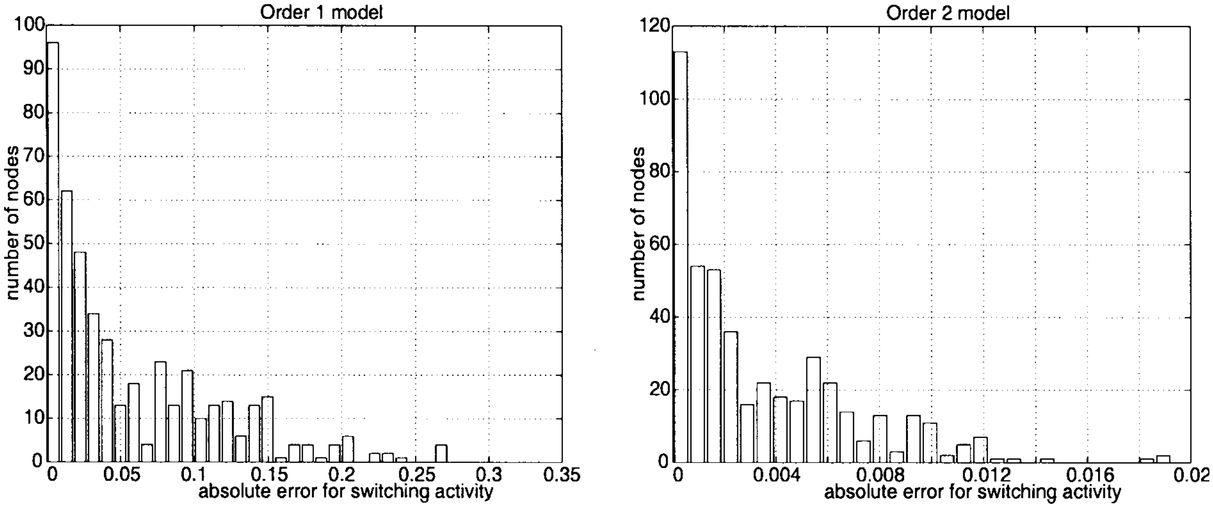
TABLE V
Total Power (μW @ 20 MHz) for Composite Input Sequences (10 000 Vectors

			Estimated power for $r = 10$			Estimated power for $r = 20$		
Circuit	No. of inp./FFs	Power for initial seq.	Order 0	Order 1	Adaptive	Order 0	Order 1	Adaptive
s1196	14/18	5272.36	7212.58	5395.73	5237.65	7245.41	5501.05	4972.52
s1423	17/74	3964.48	5771.69	4173.21	4048.88	5836.40	4046.63	3834.35
s510	19/6	1520.46	2232.08	1814.65	1527.91	2215.03	1953.39	1501.10
s5378	35/164	11566.16	14251.17	11711.97	11282.86	14325.13	11871.03	10990.86
s820	18/5	3131.51	4158.82	3368.83	3096.87	4178.75	3431.04	3196.87
s9234	36/211	10046.05	12797.24	9688.30	9573.48	12951.02	9672.07	9288.65
s953	16/29	764.92	1188.94	796.02	755.66	1186.26	805.89	731.35
		Avg. % err.	38.29	6.20	1.82	38.80	8.04	4.18
bbara	4/4	771.16	790.23	842.87	769.97	806.44	749.96	766.41
bbtas	2/3	405.29	333.21	427.52	403.45	339.82	443.14	402.84
dk17	2/3	1392.67	1165.08	1258.88	1390.21	1135.61	1248.20	1388.09
donfile	2/5	3203.97	2455.90	3100.94	3197.36	2463.37	2997.86	3190.14
mc	3/2	335.97	289.37	298.76	332.92	295.64	285.17	328.27
planet	7/6	8619.93	7753.93	7234.10	8401.28	7605.67	7097.61	8030.61
shiftreg	1/3	149.59	100.62	140.30	149.29	97.36	139.68	148.97
		Avg.% err.	16.65	8.71	0.66	17.28	9.76	1.65

the rule of Fibonacci is a second-order recursive relation, we are guaranteed that second-order temporal correlations are generated and, therefore, we can asses the effectiveness of the adaptive model. For the typical case of $s1196$, the bit-level switching activity varies in the range of 0.33–0.5 for the order two subsequences and 0.44–0.55 for the counted subsequence. In terms of power consumption, the impact of these different characteristics is illustrated by a 7025.31 μW power consumption for the corresponding order two subsequence and 2643.18 μW for the sequence of order one.

As shown in Table V, the initial sequences were compacted with two different compaction ratios (namely $r = 10$ and 20) using three Markov models: one of order zero (that is, assuming temporal independence on the primary inputs), one of order one based on DMT_1 and another one matching the actual characteristics of the sequence. This table shows the total power dissipation measured for the initial sequence (column 3) and for the compacted sequence using all models (columns 4–9). Using a Sparc 20 workstation with 64 Mbyte of memory, the time necessary to read and compress data was less than 10 s for all models. Since the compaction with DMC modeling is linear in the number of levels in the DMT_k structure, these time values are far less than the actual time needed to simulate the original sequence. During these experiments, the maximum number of nodes allowed in the Markov model was 200 000.

As we can see, the most dramatic increase in the level of error occurs for the model of order zero. This proves that the temporal independence assumption on the primary inputs impairs the accuracy of the estimation for all practical purposes. This is because for a sequence generated with a second-order source, a model that ignores temporal correlations (or considers only pairs of consecutive vectors) cannot preserve correctly even the first-order transition probabilities on the primary inputs and state lines [that is, the probabilities $p(x_n s_n x_{n-1} s_{n-1})$ in our notation]. The error for the first-order model is, on average, around 10%, while the adaptive modeling technique provides accurate results, even for a compaction ratio of $r = 20$. For example, for $s1196$ in Table V, instead of simulating 10 000 vectors with an exact



(a)

(b)

Fig. 20. Node-by-node analysis for benchmark *planet*.

power value of 5272.36 μW, one can use only 1000 vectors ($r = 10$) with an estimate of 5237.65 μW.

Regarding the results presented in Table V, we note that for order zero and order one experiments we report the best results obtained over a set of 100 runs, while for the adaptive case, we report the worst result. This way, we consider the worst-case scenario for our adaptive sequence compaction approach and ensure that even so, on average, the adaptive approach works better. The reason behind this is the fact that the spread of total power values obtained over a large number of runs for the adaptive case is typically three orders of magnitude smaller than in the case of using a fixed order one. For instance, for $s5378$, the standard deviation for the adaptive and order-one models is 1.69 and 222.61, respectively. Similarly, for the case of $s1423$ benchmark, the standard deviations are 0.34 and 240.44 for the adaptive and order-one models, respectively. Thus, it is expected that, with high confidence, the adaptive approach will work better than the fixed order-one model.

As for comparing our results with simple random sampling technique for FSM circuits, we are unable to do so for the following reason. Our compaction technique starts with a finite input sequence and a user-specified compaction ratio or error level, and performs compaction by DMC modeling and sequence generation. We must, therefore, compare our results with statistical sampling techniques which work on finite populations (i.e., an input sequence with fixed length). Although sampling techniques for combinational circuits under a given input sequence have been developed and published in the literature (hence, we could produce results of Tables II and III), such techniques for FSM circuits are not known. Note that Monte Carlo simulation (which is based on a simple random sampling strategy) assumes an infinite population and it synthesizes the input sequence used for sampling based on a Markov model of bitwise activities. Hence, it cannot be used for our comparison purpose.

Finally, we give in Fig. 20 the *node-by-node* switching activity analysis for benchmark *planet* using a second-order source on the primary inputs and a compaction ratio of five. Using a lower-order model than the actual order of the input sequence can significantly impair the ability of correctly estimating the switching activity on a node-by-node basis. While for the first-order model the absolute error[10] achieves a maximum value of 0.272 and a mean value of 0.057, it decreases to 0.019 and 0.003, respectively, if a second-order model is used. These results are typical for the whole set of benchmarks that we analyzed. Based on these results, we can, therefore, conclude that for FSM's the adaptive technique is the only technique appropriate for correctly modeling the input sequence.

In Table VI, we provide the gate-level power simulation results for a set of different initial sequences having a length of 200 000 vectors. The *input data* is set of composite sequences consisting of a first-order temporally correlated subsequence (length 100 000) followed by a second-order one. The first-order subsequences are generated with counters restarted at random after a fixed number of patterns have been generated. The second-order sequences are generated with Fibonacci time series. For the same typical example, the bit-level switching activity varies in the range of 0.33–0.5 for the order-two subsequences and 0.49–0.5 for the counted subsequence. In terms of power consumption, we have a dissipation of 4225.11 μW when the input sequence has order one and 7223.43 μW for the other one.

To generate the results in Table VI, we use two different strategies. First, we compact the input sequences with two fixed compaction ratios and indicate the estimated values of total power consumption (columns 4 and 5). Using a Sparc 20 workstation with 64 Mbyte of memory, the time necessary to read and compress data was less than 10 s in all cases. In the second scenario (columns 6 and 7), we consider a fixed threshold of 5% for the mean error of the transition probabilities. We monitor the current values of the transition probabilities and compare them with the transition

[10] The absolute error is defined as $|sw_{\text{comp}} - sw_{\text{exact}}|$, where sw_{comp} is the switching activity obtained using the compacted sequence.

TABLE VI
TOTAL POWER (μW @ 20 MHz) FOR LONG SEQUENCES (200 000 VECTORS)

			Estimated power for fixed compaction ratio		Fixed error level (5% mean error for transition probabilities)	
Circuit	No of inp./ FFs	Power for initial seq.	$r = 10$	$r = 100$	Estimated power	Number of vectors required
s1196	14/18	5724.28	5723.42	5702.51	5714.14	2242
s1423	17/74	3544.75	3548.86	3548.02	3563.84	8984
s510	19/6	1685.12	1692.89	1307.42	1690.10	8002
s5378	35/164	9161.58	9139.20	9355.99	9169.60	11884
s820	18/5	3609.23	3608.02	3604.97	3616.75	5962
s9234	36/211	9570.30	9569.93	9568.40	9577.91	23922
s953	16/29	817.91	817.60	816.81	817.42	4922
		Avg. % err.	0.12	3.62	0.20	
bbara	4/4	777.10	777.10	776.29	776.71	1404
bbtas	2/3	422.18	422.19	422.26	423.23	84
dk17	2/3	1381.27	1381.16	1380.02	1359.41	84
donfile	2/5	3250.47	3250.43	3249.96	3184.50	84
mc	3/2	346.03	346.00	345.69	335.66	162
planet	7/6	7857.02	7859.57	7880.29	7881.93	2524
shiftreg	1/3	150.86	150.85	150.76	145.93	144
		Avg. % err.	0.01	0.10	1.54	

probabilities of the original sequence. When the difference between the two sets of probabilities becomes sufficiently small, the generation procedure is halted. In this way, we are able to satisfy any user specified error level for the transition probabilities.

As we can see, the average error in total power prediction is below 2% for all benchmarks while the achieved compaction ratio varies between 8 ($s9234$) and 90 ($s1196$) for large circuits and 80 (*planet*) and 2380 (*bbtas*) for small ones. This reduction in the sequence length has a significant impact on speeding-up the simulative power estimation approaches where the running time is proportional to the length of the sequence which must be simulated.

V. CONCLUSION

In this paper, we addressed the issue of sequence compaction for power estimation from a probabilistic point of view. More precisely, we proposed an original approach to compact an initial sequence into a much shorter equivalent sequence, which can be used with any available simulator to derive power estimates in the target circuit.

A major contribution of this paper is that it introduces the hierarchical modeling of Markov chains as a flexible framework for capturing not only complex spatiotemporal correlations, but also the dynamic changes in the input sequence characteristics. In addition to this, we introduce and characterize a family of variable-order dynamic Markov models which provide an effective way for accurate modeling of external input sequences that affect the behavior of FSM's.

Results obtained on standard combinational and sequential benchmarks, show that using this framework, large compaction ratios can be obtained without significant loss in the accuracy of total and node-by-node power estimates.

REFERENCES

[1] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Trans. Design Automation Electron. Syst.,* vol. 1, no. 1, pp. 1–54, Jan. 1996.

[2] C. M. Huizer, "Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation," in *Proc. IEEE European Solid-State Circuits Conf.,* 1990, pp. 61–64.

[3] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, "The design and implementation of PowerMill," in *Proc. Int. Workshop Low-Power Design,* Apr. 1995, pp. 105–110.

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

[5] C.-S. Ding, C.-T. Hsieh, Q. Wu, and M. Pedram, "Stratified random sampling for power estimation," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design,* San Jose, CA, Nov. 1996, pp. 577–582.

[6] T. L. Chou and K. Roy, "Statistical estimation of sequential circuit activity," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design,* Nov. 1995, pp. 34–37.

[7] F. Najm, S. Goel, and I. Hajj, "Power estimation in sequential circuits," in *Proc. ACM/IEEE Design Automation Conf.,* June 1995, pp. 635–640.

[8] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proc. ACM/IEEE Design Automation Conf.,* June 1992, pp. 253–259.

[9] F. N. Najm, "Transition density: A new measure of activity in digital circuits," in *IEEE Trans. Computer-Aided Design,* vol. 12, pp. 310–323, Feb. 1993.

[10] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Efficient estimation of dynamic power dissipation with an application," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design,* San Jose, CA, Nov. 1993.

[11] R. Marculescu, D. Marculescu, and M. Pedram, "Probabilistic modeling of dependencies during switching activity analysis," *IEEE Trans. Computer-Aided Design,* vol. 17, pp. 73–83, Feb. 1998.

[12] C.-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, and A. M. Despain, "Power estimation in sequential logic circuits," *IEEE Trans. VLSI Syst.,* vol. 3, pp. 404–416, Sept. 1995.

[13] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient power estimation for highly correlated input streams," in *Proc. ACM/IEEE Design Automation Conf.,* June 1995, pp. 628–634.

[14] D. Marculescu, R. Marculescu, and M. Pedram, "Stochastics sequential machine synthesis targeting constrained sequence generation," in *Proc. 33rd ACM/IEEE Design Automation Conf.,* Las Vegas, NV, June 1996.

[15] R. Marculescu, D. Marculescu, and M. Pedram, "Vector compaction using dynamic Markov models," in *IEICE Trans. Fundamentals,* Japan, Oct. 1997, vol. E80-A, no. 10, pp. 1924–1933.

[16] G. V. Cormack and R. N. Horspool, "Data compression using dynamic Markov modeling," *Comput. J.,* vol. 30, no. 6, pp. 541–550, 1987.

[17] C.-Y. Tsui, R. Marculescu, D. Marculescu, and M. Pedram, "Improving the efficiency of power simulators by input vector compaction," in *Proc. 33rd ACM/IEEE Design Automation Conf.,* Las Vegas, NV, June 1996.

[18] G. Kemeny, J. L. Snell, and A. W. Knapp, *Denumerable Markov Chains.* D. Van Nostrand Company, Inc., 1966.

[19] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* New York: McGraw Hill, 1984.

[20] E. Seneta, "Non-Negative Matrices." New York: Wiley, 1973.

[21] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* Oxford, U.K.: Clarendon, 1988.

[22] J. W. Green and K. J. Supowit, "Simulated annealing without rejected moves," in *Dig. Int. Conf. Computer Design,* Oct. 1984, pp. 658–663.

[23] A. I. Khinchin, *Mathematical Foundations of Information Theory.* New York: Dover, 1957.

[24] A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Stream synthesis for efficient power simulation based on spectral transforms," in *Proc. Int. Symp. Low-Power Electronic Design,* Aug. 1998, pp. 30–35.

[25] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation, and optimization," *IEEE Trans. Computer-Aided Design,* vol. 17, pp. 1061–1079, Nov. 1998.

Radu Marculescu (S'94–M'98) received the M.S. degree in electrical engineering from the Technical University of Iasi, Iasi, Romania, in 1985 and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1998.

Currently, he is an Assistant Professor in the Electrical and Computer Engineering Department at the University of Minnesota, Minneapolis. His research interests include CAD of VLSI with emphasis on low-power design methodologies.

Diana Marculescu (S'94–M'98) received the M.S. degree in computer science from the Technical University of Bucharest, Bucharest, Romania, in 1991 and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, in 1998.

She is currently an Assistant Professor in the Electrical Engineering Department at the University of Maryland, College Park. Her research interests include hardware and software issues in low-power systems and embedded systems design.

Massoud Pedram (S'88–M'90–SM'98) received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, in 1986 and the M.S. and Ph.D. degrees from University of California, Berkeley, in 1989 and 1991, respectively.

Currently, he is an Associate Professor of Electrical Engineering at the University of Southern California, Los Angeles. His research interests span many aspects of design and synthesis of VLSI circuits, with particular emphasis on layout-driven synthesis and design for low-power.