FSM Analysis Using High-Order Markov Modes

Diana Marculescu, Radu Marculescu
and Massoud Pedram

CENG 97-08

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213 740-4458)

January 1997

FSM Analysis Using High-Order Markov Models

Abstract

Accurate evaluation of steady-state and transition probabilities is an essential step in analyzing the behavior of finite-state machines (FSMs). Recently, concepts from Markov chain theory have been successfully applied to probabilistically model this behavior. The objective of this paper is to investigate the effect of finite-order statistics of the external input sequences on the behavior of FSMs and, more generally, of a network of interacting FSMs. Our proposed approach is probabilistic in nature and relies on adaptive modeling of binary input streams as fixed-order Markov sources of information. This approach is very effective and flexible: the input-modeling Markov model itself is derived through a one-pass traversal of the input sequence and can be excited to generate an equivalent sequence, much shorter in length compared to the original sequence. The compressed sequence, can be subsequently used with any available simulator to derive the steady-state and transition probabilities, and the total power consumption in the target circuit. As the results demonstrate, large compaction ratios of orders of magnitude can be obtained in a matter of few seconds without significant loss (less than 3% on average) in the accuracy of estimated values.

1. Introduction

In the last decade, probabilistic approaches have received a lot of attention as a viable alternative to deterministic techniques for analyzing complex digital systems. Logic synthesis [1], verification [2], testing [3][4] and more recently, low-power design [5] have benefited from using probabilistic techniques. In particular, the behavior of FSMs has been investigated using concepts from the Markov chain theory. In fact, the use of discrete Markov models is almost natural: simply relabel the out-going edges that correspond to each particular state in the state transition graph (STG) with the probability for the FSM to make that particular transition, to obtain a finite state model that matches the requirements of the Markov chains [6].

Studying then the behavior of the Markov chain provides us with different variables of interest of the original FSM. In this direction [7][8] are excellent references where steady-state and transition probabilities (as variables of interest) can be successfully estimated in large FSMs. However, both techniques are analytical in nature and, in order to manage complexity, have to consider some simplifying assumptions. As a consequence, only logic simulation of the actual set of inputs can finally assert the accuracy of results. (In fact, verification through logic simulation, will persist as the final step to decide the quality of results provided by any analytical approach.) It is, however, impractical to simulate long sequences of vectors, mostly when the target circuit is large or when many runs are needed to evaluate a number of alternative designs. From this perspective, a short/compact sequence of stimuli - which is representative of the typical application data - would be desirable to speed-up the simulation. Differently stated, the question to be answered is: having a sequence S_1, assumed representative of the data applied to a target sequential circuit, can we produce a shorter sequence S_2 such that the steady-state and transition probabilities of the signal lines are nearly preserved?

The aim of this paper is to address this issue and, based on a new Markov model, to propose an effective way to solve it not only for standard FSMs but also for interacting FSMs. The knowledge of steady-state and transition probabilities is a very important topic by itself because both of them completely characterize the FSM behavior. However, as a particular domain where they have an immediate application, we chose the power estimation area. Without loss of generality, we will consequently emphasize the applicability of the new results on sequence compaction for power estimation. The reason for doing so is three fold: first, because most of the prior work that has extensively used Markov chain modeling for FSMs was also directed towards solving the power estimation problem; second, because in power estimation (which is an important topic nowadays) accurate estimation of transition probabilities is a key factor as the total power consumption is highly sensitive to transition probability changes [9]; third, for convenience in reporting the results (there is one power value per circuit, but 2^{2k} one-step transition probabilities for any k signal lines taken into consideration).

Generating a minimal-length sequence of input vectors that satisfies a prescribed set of statistics in not a trivial task. More precisely, LFSRs which have traditionally found use in testing or functional verification [4], are of little or no help here. The reason is that more elaborate set of input statistics must be preserved or reproduced during sequence generation for use by power simulators. One such attempt is [10] where authors use deterministic FSMs to model user-specified input sequences. Since the number of states in the FSM is equal to the length of the sequence to be modeled, the ability to characterize anything else but short input sequences is limited. More elaborate and effective techniques were recently presented in [11] [12] where the authors succeed in compacting large sequences with very small loss in accuracy. However, these approaches are suited only for combinational circuits; this is because both of them consider only first-order temporal effects, that is they analyze only pairs of consecutive vectors to perform sequence compaction. As we will prove in this paper, in the case of FSMs, this is not enough for accurate estimation of transition probabilities. Temporal correlations longer than one time step can affect the overall behavior of the FSM and therefore, result in very different power consumptions. Let us illustrate this point using a simple example.

Example 1: Let S_1 and S_2 be two 4-bit sequences, of length 26, as shown in Fig.1a. 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.1b 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, $v_1 = v_2 =$ '1', $v_3 =$ '2',..., $v_{26} =$ '9'). Each edge represents a valid transition between any two valid patterns and has associated a nonzero probability with it. For instance, the pattern '13' in S_1 and S_2 is always followed by '5' (then the edge between nodes '13' and '5' has the probability 1) while after pattern '2' is equally likely to follow either '3' or '7'.

S_1

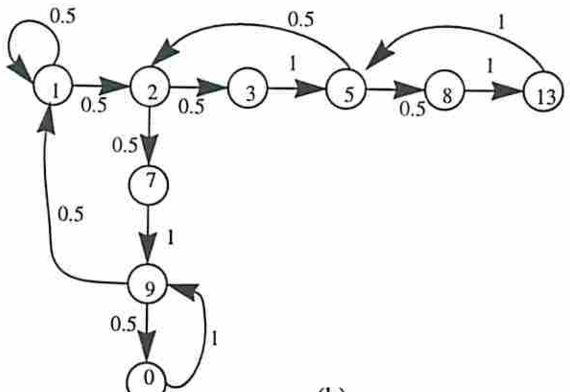
v_{26} ... $v_3 v_2 v_1$

10100011000001010001100000
00010110100000001011010000
00011000011000001100001100
10110110110111011011011011

S_2

10100011000011000001010000
00010110100110100000001000
00011000011000011000001100
10110110110110110111011011

(a)



(b)

Fig.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 see then the graph in Fig.1b as a compact, canonical, characterization of sequences S_1 and S_2. Suppose now that S_1 and S_2 feed in turn the benchmark *s8* taken from the *mcnc*91 sequential suite. Looking at different internal nodes of the circuit, we can see 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 384uW and 476uW, respectively, showing a difference of more than 24% even for this small set of inputs. This difference may go even worse; for instance, the benchmark *dk512* exhibits a difference of 32% in the total power consumption when S_1 and S_2 are applied in turn at its primary inputs.

A natural question is then, why this difference is appearing, despite the fact that S_1 and S_2 have the same characteristic graph plotted in Fig.1b. The reason resides in the fact that S_1 and S_2 have a different set of second-order statistics that is, the sets of triplets (three consecutive patterns) are different. For instance, the triplet (1,2,7) in S_2 is never occurring in S_1 (because in S_1 the pattern '5' is always the ancestor of the vector pair (2,7)); the same observation applies to the triplet (5,2,3) in S_2. The conclusion to note is that having the same sets of one-step transition probabilities *does not* mean that the sets of second-(or higher) order statistics are identical and, as it was just illustrated in this small example, for FSMs higher order statistics can make a significant difference in total power consumption. The initial problem of producing a shorter sequence (compared to a given one) which can preserve the set of steady-state and transition probabilities can be casted now in terms of power as follows: can we transform a given input sequence into a smaller one, such that the new body of data represents a good approximation as far as total power consumption is concerned?

Addressing these issues, the present paper improves the state-of-the-art in two ways: first, it shows the effect of finite-order statistics of the input sequence on FSMs behavior; second, based on the vector compaction paradigm, it provides an original solution for power estimation problem in FSMs and interacting FSMs. From the original results provided here, two are noteworthy for probabilistic FSM analysis:

- if the sequence feeding the target circuit has order k, then a lag-k Markov chain model of the sequence will suffice to model correctly the joint transition probabilities of the primary inputs and internal states in the target circuit;
- if the input sequence has order two or higher then modeling it as a lag-one Markov Chain cannot exactly preserve the first-order joint transition probabilities (primary inputs and internal states) in the target circuit.

The foundation of our approach is probabilistic in nature; it relies on *adaptive* (*dynamic*) *modeling* of binary input streams as first- and higher-order Markov sources of information. The adaptive modeling technique itself (best known as Dynamic Markov Chain or DMC modeling) was introduced very recently in the literature on data compression [13] as a candidate to solve various data compression problems. From the very beginning, this technique looked very promising and indeed, in most practical situations, has been more effective than any other compression technique available to date. However, the original model introduced in [14] is not completely satisfactory for our purpose; in order to capture high-order temporal effects, we extend the initial formulation to handle groups of k consecutive input vectors.

To conclude, from this research may benefit not only simulation-based and probabilistic approaches for power estimation but also the (general) FSM analysis techniques relying on probabilistic premises. The issues brought into attention in this paper are new and represent an important step towards understanding of FSMs behavior from a probabilistic point of view. Finally, the main results presented here may find useful applications in other CAD applications.

The paper is organized as follows: based on Markovian information sources, in Section 2 we present the main results about the effect of finite-order statistics on FSM and interacting FSM behavior. Section 3 formalizes the power-oriented vector compaction problem and introduces a DMC-based procedure for vector compaction. In sections 4 and 5 we give some practical considerations and experimental results, respectively. Finally, we conclude by summarizing our main contribution.

2. Markovian sources of information

In what follows, we characterize the input sequences as binary information sources of discrete Markov type that emit an input vector at every time step. For all practical purposes this is motivated by the following two paradigms [15]:

- first, the dependence of some model of interest on its past values may be non-Markovian but still be based on a finite-order memory;
- second, there are so-called *embedded regeneration points*; these are times at which the system forgets its past in a probabilistic sense: the system viewed at such time points is Markovian even if the overall process is not.

2.1 Finite-order memory models

We focus first on the input sequence that supposedly feeds a target circuit and we model it as an information source. We consider therefore the *model* as having two parts: 1) the *structure* which is the set of events and their contexts (the set of bits or words surrounding some bit or word under consideration) and 2) the *parameters* which are probabilities assigned to the events. The structure is the same for the entire set of sequences under consideration while the parameters are tailored to each individual sequence. Without loss of generality, we restrict ourselves to finite binary strings, that is, finite sequences consisting only of 0's and 1's. The set of events of interest is the set S of all finite binary sequences on b bits. A particular sequence S_1 in S consists of vectors $v_1, v_2, \ldots, v_n$ (which may be distinct or not), each having a positive occurrence probability[1]. Indices 1, 2,..., n represent the discrete time steps when a particular vector occurs in sequence and it is applied to a target circuit. An attractive subclass of information sources is the class of Markov sources [16] which can be conveniently modeled as Markov chains of finite-order.

Definition 1. (lag-k Markov chain) A discrete stochastic process $\{v_n\}_{n \geq 1}$ is said to be a lag-k Markov chain if at any time step n:

$$p(v_n | v_{n-1} v_{n-2} \ldots v_0) = p(v_n | v_{n-1} v_{n-2} \ldots v_{n-k}) \tag{1}$$

In particular, any lag-one Markov source, is characterized by the set of internal states (nodes in the corresponding graph representation) and a set of transition probabilities p_{ij} that give the transition probability from state v_i to the next state v_j. It should be noted that any lag-k Markov chain can be reduced to a lag-one Markov chain based on the following result.

Proposition 1. If $\{u_n\}_{n \geq 1}$ is a lag-k Markov chain then $\{v_n\}_{n \geq 1}$ where $v_n = (u_n, u_{n-1}, \ldots, u_{n-k+1})$ is a multivariate first-order Markov chain.

We also have the following proposition:

Proposition 2 [6]. The probability of occurrence for a vector string $v = v_1 v_2 \ldots v_n$ is given by:

$$p(v) = p(v_1) \cdot p(v_2 | v_1) \cdot \ldots \cdot p(v_n | v_1 v_2 \ldots v_{n-1}) \tag{2}$$

where the conditional probabilities are uniquely defined by: $p(x|v) = p(vx) / p(v)$.

<u>Example 2</u>: Assume as given the sequences S_1 and S_2 in Introduction; we want to compute the probability of occurrence of the string v = '0101 0010' that is, the probability that transition $5 \rightarrow 2$ is taking place in both sequences. To this effect, we just apply formula (2) and then $p(v) = p(v_1 v_2) = p(v_1) \cdot p(v_2 | v_1)$ which gives us the value of 1/13 in both cases. On the other side, if we are interested in the two-step transition $5 \rightarrow 2 \rightarrow 7$ then applying once again (2) we get the values 1/13 and 1/26 for S_1 and S_2, respectively. This difference shows that despite the fact that S_1 and S_2 have the same set of first-order transition probabilities, they do have different second-order statistics.

2.2 The effect of finite-order statistics of the input sequence on FSMs behavior

Now we turn our attention from the input sequence to the circuit and we investigate the effect of input statistics on the transition probabilities (primary inputs and present state lines) in the target circuit. As shown in Fig.2, we model the 'tuple' (*input_sequence, target_circuit*) by the 'tuple' (*Markov_chain, target_circuit*), where *Markov_chain* models the *input_sequence* and *target_circuit* is the sequential machine where the transition probabilities have to be determined. In what follows, x_n, s_n will denote the inputs and states of the target sequential machine; $p(x_n s_n)$ is the probability that the input is x_n and the state is s_n at time step 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 we can see in Fig.2, they 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.

---

1. Throughout the paper, we may refer occasionally to vectors $v_1, v_2, \ldots, v_n$ as 'symbols', 'patterns' or 'states'.

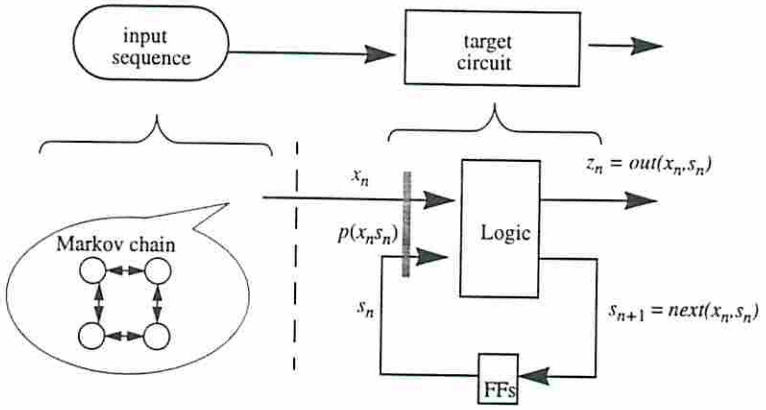


Fig.2

Theorem 1. If input x_n applied to a target sequential circuit can be modeled by a lag-k Markov chain then, for any $n \geq 1$, the following holds:

$$p(x_n s_{n-k} | x_{n-1} x_{n-2} \dots x_{n-k}) = p(x_n | x_{n-1} x_{n-2} \dots x_{n-k}) \cdot p(s_{n-k} | x_{n-1} x_{n-2} \dots x_{n-k}) \qquad (3)$$

Proof: By definition, x_n is a lag-k Markov chain if and only if $p(x_n | x_{n-1} x_{n-2} \dots x_{n-p}) = p(x_n | x_{n-1} x_{n-2} \dots x_{n-k})$ for every $p > k$. Hence, x_n is independent on any x_{n-p} with $p > k$ and depends only on $x_{n-1}, x_{n-2}, \dots, x_{n-k}$. On the other hand, s_{n-k} is a function only of s_{n-k-1} and x_{n-k-1}. Thus, if $x_{n-1}, x_{n-2}, \dots, x_{n-k}$ are known, then x_n and s_{n-k} are independent which is exactly (3). In other words, x_n and s_{n-k} are conditionally independent[1] with respect to $x_{n-1} \dots x_{n-k}$. ■

Theorem 2. If the sequence feeding a target sequential circuit has order k, then a lag-k Markov chain will suffice to model correctly the k-step conditional probabilities of the primary inputs and internal states in the target circuit, that is $p(x_n s_n | x_{n-1} s_{n-1} x_{n-2} s_{n-2} \dots x_{n-k} s_{n-k}) = p(x_n | x_{n-1} x_{n-2} \dots x_{n-k})$.

Proof: Let $p(x_n s_n x_{n-1} s_{n-1} \dots x_{n-k} s_{n-k})$ be the joint transition probability for inputs and states at time step n. Then we have:

$$p(x_n s_n \dots x_{n-k} s_{n-k}) = \begin{cases} p(x_n x_{n-1} \dots x_{n-k} s_{n-k}) & \text{if } next(x_i, s_i) = s_{i+1} \quad i = n-k \dots n-1 \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

For the first alternative we have:

$$p(x_n s_n \dots x_{n-k} s_{n-k}) = p(x_n x_{n-1} \dots x_{n-k} s_{n-k}) = p(x_n s_{n-k} | x_n x_{n-1} \dots x_{n-k}) \cdot p(x_n x_{n-1} \dots x_{n-k}) \qquad (5)$$

Since x_n is a lag-k Markov chain, from Theorem 1 we get:

$p(x_n s_{n-k} | x_{n-1} x_{n-2} \dots x_{n-k}) = p(x_n | x_{n-1} x_{n-2} \dots x_{n-k}) \cdot p(s_{n-k} | x_{n-1} x_{n-2} \dots x_{n-k})$ or equivalently, using (4), we obtain: $p(x_n s_n \dots x_{n-k} s_{n-k}) = p(x_n | x_{n-1} x_{n-2} \dots x_{n-k}) \cdot p(x_{n-1} x_{n-2} \dots x_{n-k} s_{n-k})$.

Dividing both sides by $p(x_{n-1} x_{n-2} \dots x_{n-k} s_{n-k})$ and using the second part of (5) we obtain exactly $p(x_n s_n | x_{n-1} s_{n-1} x_{n-2} s_{n-2} \dots x_{n-k} s_{n-k}) = p(x_n | x_{n-1} x_{n-2} \dots x_{n-k})$. For the second alternative, if $x_n s_n x_{n-1} s_{n-1} x_{n-2} s_{n-2} \dots x_{n-k} s_{n-k}$ is not a valid sequence,

1. The concept of conditional independence is discussed in [6].

then $p(x_n s_n | x_{n-1} s_{n-1} x_{n-2} s_{n-2} \ldots x_{n-k} s_{n-k}) = 0$ and this concludes our proof. ■

We note therefore that preserving order-k statistics implies also that order-one statistics will be captured for inputs and states.

Corollary 1. If the sequence feeding the target circuit has order one, then a lag-one Markov chain will suffice to model correctly the joint transition probabilities of the primary inputs and internal states in the target circuit, that is $p(x_n s_n | x_{n-1} s_{n-1}) = p(x_n | x_{n-1})$.

Thus, if x_n is a lag-one Markov chain, preserving the first-order statistics on the inputs is enough for preserving the joint transition probabilities for inputs and states. In other words, if the sequence feeding a 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 only to two consecutive time steps.

Theorem 3. If the input sequence has order two, then modeling it as a lag-one Markov Chain will *not* preserve exactly the first-order joint transition probabilities (primary inputs and internal states) in the target sequential circuit.

Proof: Based on (4) for $k = 2$ we have:

$$p(x_n s_n x_{n-1} s_{n-1}) = p(x_n x_{n-1} s_{n-1}) \qquad \text{or} \qquad p(x_n s_n | x_{n-1} s_{n-1}) = p(x_n | x_{n-1} s_{n-1})$$

for all valid combinations of inputs and states (otherwise is zero).

Let's assume that the input is modeled as a lag-one Markov chain. Thus, its first order statistics are preserved. Let p' be the new probability distribution under this assumption. We have:

$$p'(x_n | x_{n-1}) = p(x_n | x_{n-1}) \qquad \text{but} \qquad p'(x_n | x_{n-1} x_{n-2}) \neq p(x_n | x_{n-1} x_{n-2}).$$

Using Corollary 1, the one-step conditional probability of the inputs and states becomes:

$$p'(x_n s_n | x_{n-1} s_{n-1}) = p'(x_n | x_{n-1}) = p(x_n | x_{n-1})$$

Assuming by contradiction that the first order statistics are also preserved jointly for inputs and states, using Theorem 1 we obtain:

$$p'(x_n s_n | x_{n-1} s_{n-1}) = p(x_n s_n | x_{n-1} s_{n-1}) \Leftrightarrow p(x_n | x_{n-1}) = p(x_n | x_{n-1} s_{n-1}) \Leftrightarrow$$

$$p(x_n | x_{n-1}) = \frac{\displaystyle\sum_{\substack{x_{n-2}, s_{n-2} \\ next(x_{n-2}, s_{n-2}) = s_{n-1}}} p(x_n x_{n-1} x_{n-2} s_{n-2})}{\displaystyle\sum_{\substack{x_{n-2}, s_{n-2} \\ next(x_{n-2}, s_{n-2}) = s_{n-1}}} p(x_{n-1} x_{n-2} s_{n-2})} \Leftrightarrow$$

$$\sum_{\substack{x_{n-2}, s_{n-2} \\ next(x_{n-2}, s_{n-2}) = s_{n-1}}} p(x_{n-1} x_{n-2} s_{n-2}) \cdot [p(x_n | x_{n-1}) - p(x_n | x_{n-1} x_{n-2})] = 0$$

Since we chose only the combinations that may appear for inputs and states, the above sum can be zero only if $p(x_n | x_{n-1}) = p(x_n | x_{n-1} x_{n-2})$. But this would mean that the input is a lag-one, whereas in reality is a lag-two Markov chain. Thus, first order statistics cannot be preserved by considering only one-step conditional probabilities on the input.■

The above result can be also generalized for k-lag Markov chains. So in general, modeling a k-order source with a lower order model may introduce accumulative inaccuracies. From a practical point of view this means that if one underestimates a high-order source (for instance, assuming that second- or higher-order temporal correlations affect only two consecutive time steps), then one may end up not preserving correctly even the first-order transition probabilities. In terms of power consumption, this will adversely affect the quality of the results. However, we will show later that increasing the order of the input model will decrease the error in correctly capturing the joint transition probabilities for inputs and states.

2.3. Interacting FSM and high-order information sources

Modern designs where interacting finite state machines are present offer a good example where high-order information sources found applicability. As presented in [17], the decomposition of large FSMs into smaller, interacting FSMs may be useful for both area and performance reasons. In practice, three options are available (Fig.3): *parallel decomposition* (both submachines are supplied with the same input sequence, but operate independently), *cascade decomposition* (one submachine has information about the internal state of the another one) and finally, a type of *complex decomposition* where each submachine is provided with information about the current state of the other submachine.

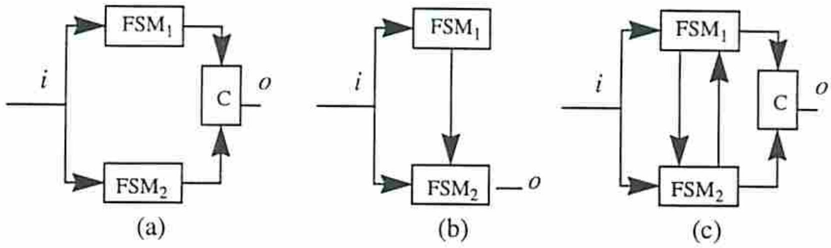


Fig.3

Starting on i with a Markov source of order k, any of the topologies in Fig.3 may increase the order of the source at the output o. This should be apparent from the following result which uses the basic notations in Fig.2.

Theorem 4.[18] If the input x_n is driven by a Markov source of order one then the output of the source z_n remains a lag-one Markov chain if and only if:

(a) the *out* function is a one-to-one mapping or

(b) the transition matrix P of the tuple $(x_n s_n)$ is of the form $P = \alpha I + (1-\alpha)U$ where U is a matrix with identical rows and α is a positive real number less than one.

Using Proposition 1, this important result can be easily extended to any lag-k Markov chain. Any of the conditions required by Theorem 4 is in fact very restrictive. As a consequence, it is expected that the order of the output z_n will increase since conditions (a) and (b) can be barely satisfied in practice. At first, this conclusion may appear disappointing, especially from a practical point of view. However, we may assume a finite-order Markov source, since for a given level of accuracy, the following general result *guarantees* the existence of a *finite limit* in the increase of the order.

Theorem 5. [19] Let $P = (p_{ij})_{1 \le i, j \le N}$ be the transition probability matrix of a lag-one Markov chain $\{x_n\}_{n \ge 1}$ with N states. If $p_{ij} > 0$ for any i, j and $\lambda = min_{i, j, k, l} \frac{p_{kj} \cdot p_{il}}{N^2 \cdot p_{ij} \cdot p_{kl}}$, then for any arbitrary function $z_n = f(x_n)$ the following holds for every k and $x_{n-k-1} \ne x'_{n-k-1}$[1]:

$$|p(z_n|z_{n-1}\ldots z_{n-k}x_{n-k-1}) - p(z_n|z_{n-1}\ldots z_{n-k}x'_{n-k-1})| \le (1-\lambda)^k . \quad (6)$$

1. It can also be shown that λ is less than one. The result may be extended to Markov chains of order greater than one.

In other words, this theorem states that even if the output is not of finite order, it can be approximated as such up to a bounded error. For example, considering a two-state Markov chain $\{x_n\}_{n \geq 1}$ where $p_{00} = 1/3$, $p_{01} = 2/3$, $p_{10} = 1/2$, $p_{11} = 1/2$, we can easily calculate $\lambda = 1/8$. For a given value of accuracy, say $\varepsilon = 0.1$, a Markov chain of order $log_{(1-\lambda)}\varepsilon = 18$ will ensure that for any $z_n = f(x_n)$, the error in (6) is less than ε.

Corollary 2.

Assume that the input of the FSM can be written as $x_n = f(w_n)$ where f is an arbitrary function and $\{w_n\}_{n \geq 1}$ is a lag-one Markov chain. If the order of the Markov model used to represent the input is increased, then the error for estimating the joint transition probabilities for inputs and states decreases.

Hint for proof: Based on the result provided by Theorem 5, it's easy to show that $\{x_n\}_{n \geq 1}$ satisfies:

$$\left|p(x_n | x_{n-1} \ldots x_{n-k} x_{n-k-1}) - p(x_n | x_{n-1} \ldots x_{n-k} x'_{n-k-1})\right| \leq (1-\lambda)^k \text{ or, even further}$$

$$\left|p(x_n | x_{n-1} \ldots x_{n-k} x_{n-k-1}) - p(x_n | x_{n-1} \ldots x_{n-k})\right| \leq (1-\lambda)^k .$$

Thus, the error of using a finite-order model for a non-finite order discrete process decreases exponentially with the order used. Hence, the larger the order, the better we approximate the model on the input and also the joint transition probabilities for inputs and states.■

Using the synthesis procedure presented in [11], a variety of Markov sources having memory of order one or higher can be constructed. As indicated by Theorem 4, any increase in memory order makes the source more sophisticated in the sense that long-range temporal correlations will affect the ability to predict the 'next' vector v_{n+1}. The important point here is that having in mind all these finite-order memory effects, it makes sense to talk about the concept of *high-order information sources* for power estimation either in sequential or interacting FSM machines.

3. Data compaction for power estimation

In the previous sections, we have defined and characterized the effect of finite-order statistics on FSM behavior. Based on results presented so far, in the present section, we investigate an efficient way to solve the data compaction problem for power estimation in sequential machines.

3.1 Problem formulation

Assuming that a gate level implementation is available, to estimate the total power dissipation, one can sum over all the gates in the circuit the average power dissipation due to the capacitive switching currents, that is: $P_{avg} = \frac{f_{clk}}{2} \cdot V_{DD}^2 \cdot \sum_n (C_n \cdot sw_n)$ where f_{clk} is the clock frequency, V_{DD} is the supply voltage, C_n and sw_n are the capacitance and the average switching activity of gate n, respectively. From here, the average switching activity per node (gate) is the key parameter that needs to be correctly determined, mostly if we are interested in a node-by-node basis power estimation. However, this parameter is highly sensitive to the input statistics, namely it depends significantly on transition probabilities among different signal lines. As shown in the previous section, high-order information sources make a significant difference in power consumption for sequential machines.

Having these issues in mind, the *vector compaction problem* for FSMs can be formulated as follows: for a k-bit sequence of length n (consisting of vectors $v_1, v_2, \ldots, v_n$), find another sequence of length $m \ll n$ (consisting of the subset $u_1, u_2, \ldots, u_m$ of the initial sequence), such that the average joint transition probability on the primary inputs and present state lines is preserved *wordwise*, for k consecutive time steps. More formally, for any two generic inputs v and u

(seen as a collection of bits) in the original and in the compacted sequence, respectively, the following holds:

$$\left| p(x_n s_n = \alpha_1 \wedge x_{n-1} s_{n-1} = \alpha_2 \wedge \ldots \wedge x_{n-k} s_{n-k} = \alpha_k) - \right.$$
$$\left. - \; p(x_n s_n = \beta_1 \wedge x_{n-1} s_{n-1} = \beta_2 \wedge \ldots \wedge x_{n-k} s_{n-k} = \beta_k) \right| < \varepsilon \qquad (7)$$

In relation (6), $\alpha_1, \alpha_2, \ldots, \alpha_k$ and $\beta_1, \beta_2, \ldots, \beta_k$ are any possible patterns obtained jointly on primary inputs and present state lines when vectors $v_1, v_2, \ldots, v_n$ ($u_1, u_2, \ldots, u_m$) from the original (compacted) sequence are applied to the primary inputs at time instances 1, 2,..., k. 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 k consecutive time steps. Before going further, we note the particular case when $k = 2$, which is the theoretical basis of vector compaction techniques recently published [11][12].

3.2. A practical solution: high-order dynamic Markov models

Let's consider now a generic sequence in S consisting of vectors $v_1, v_2, \ldots, v_n$. Imposing a total ordering among bits, such a sequence may be conveniently viewed as a binary tree (let's call it DMT_0 from *Dynamic Markov Tree of order zero*) where nodes at level j correspond to bit j ($1 \le j \le k$) in the original sequence; each edge that emerges from a node is labelled with a positive count (and therefore with a positive probability) that indicates how many times the substring from the root to that particular node, occurred in the original sequence. For clarity, let's consider the following example.

Example 3: For the sequence S_1 (in introductory part) consisting of 9 non-distinct vectors the construction of the tree DMT_0 is shown step-by-step in Fig.4a. Obviously, the whole Markov tree that models this sequence must have four levels because the original sequence is a 4-bit sequence. Without any loss in generality, we assume a left-to-right order among bits that is, the leftmost bit in any vector v_1 to v_{26} is considered as being bit number one (and consequently represented at level one in DMT_0 as shown in Fig.4a), the next bit is considered as being bit number two and so on. Every time when a vector is completely scanned (that corresponds to reaching the level four in the tree), we come back to the root and start again with the next vector in the sequence. While the input sequence is scanned, the actual counts on the edges are dynamically updated (as shown in Fig.4a for the first three vectors) such that, for this particular example, they finally become as indicated in Fig.4b. The Markov tree in Fig.4b contains in a compact form all the spatial information about the original sequence $v_1, v_2, \ldots, v_{26}$.

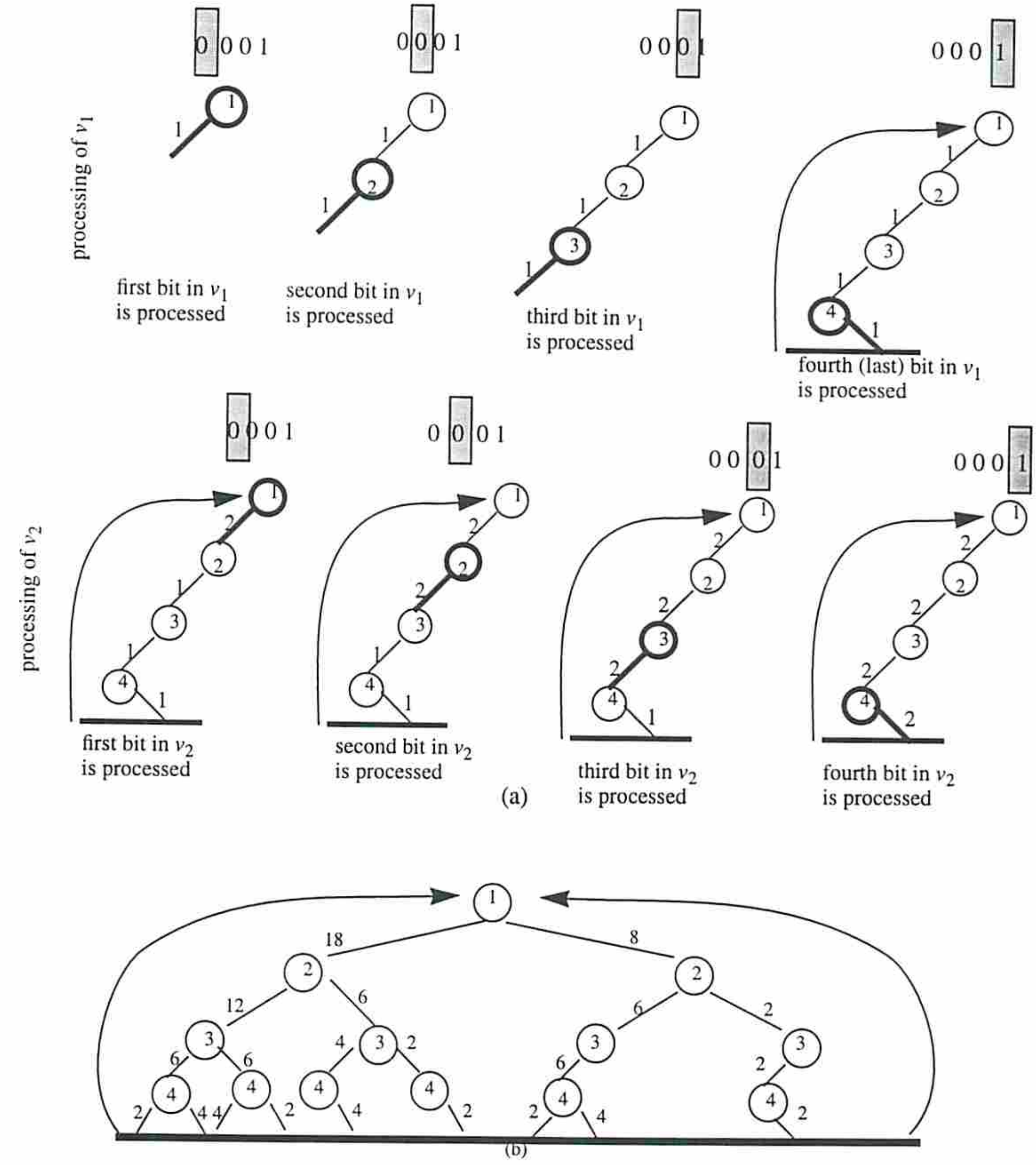


Fig.4

Proposition 3. At every node in DMT_0 we have:

$$p(v) = \sum_{u \in S} p(vu) \tag{8}$$

for all v in S, where vu represents the event corresponding to the joint occurrence of the strings v and u.

The above condition, simply states that the sum of the counts attached to the immediate successors of node v equals its own value $p(v)$[1]. In addition, based on the counts of the terminal edges, we may easily compute the probability of occurrence for a particular vector in the sequence. For instance, the probability of occurrence for string

1. This is actually similar to Kirchoff's law for currents.

'0010' is 4/26 (because the count on the terminal edge that corresponds to '0010' is 4 and the length of the sequence is 26) while the probability of string '1111' is zero, '1111' being a vector which does not occur in this particular sequence.

We observe that DMT_0 alone has *no memory* and therefore it does not preserve properly the order of events. In Fig.4b for instance, the value of 2/13 is the probability to see the particular string (state) '0010' in the original sequence but this gives us no indication at all about the sequencing of this vector relative to another one, say '0001'. As follows from the previous section, the temporal correlations that characterize a particular sequence are the key factor in power estimation. More precisely, different interleavings among the vectors belonging to the same initial set $(v_1, v_2,..., v_n)$ (e.g.$(v_1,...,v_i, v_j, v_k,...,v_n)$, $(v_1,...,v_i, v_k, v_j,...,v_n)$ or $(v_1,...,v_k, v_i, v_j,...,v_n)$) define completely different input sequences.

Consequently, in order to solve properly the compaction problem, we refine now the above structure by incorporating in it *first-order memory effects*. Specifically, we consider a more intricate structure, namely a tree called DMT_1 (*Dynamic Markov Tree of order 1*), where from the node representing any vector v there is an emergent arc to each value x connecting v to the successor node, associated with the string vx.

Example 4: For the following 3-bit sequence, consisting of 17 non-distinct vectors: $(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, v_{17})$ = (001, 100, 001, 110, 111, 111, 101, 110, 011, 000, 101, 001, 100, 000, 110, 110, 011), suppose we want to construct its corresponding tree DMT_1. We begin as we did with DMT_0 and for each leaf that represents a valid combination in the original sequence, we construct a new tree (having the same depth as DMT_0) which is meant to preserve the *context* in which the next combination occurs. For instance, the vector $v_2 = 100$ follows immediately after $v_1 = 001$; consequently when we reach the node that corresponds to v_1 (the leftmost path in Fig.5a), instead of going back to the root (and therefore 'forgetting' the context), we start to build a new tree (rooted at the current leave of DMT_0) as indicated in Fig.5a. The newly constructed tree will preserve the context in which v_2 = 100 occurred that is, immediately after $v_1 = 001$ (denoted by $v_1 \rightarrow v_2$). After processing the pair (v_1,v_2), we come back to the root and continue with (v_2,v_3) as shown in Fig.5b; v_2 alone leads us to the second leftmost edge of DMT_0 from where, to construct DMT_1, we have to add the path '001' which corresponds to v_3. In this way, we indicate the sequencing between v_2 and v_3 that is, $v_2 \rightarrow v_3$.

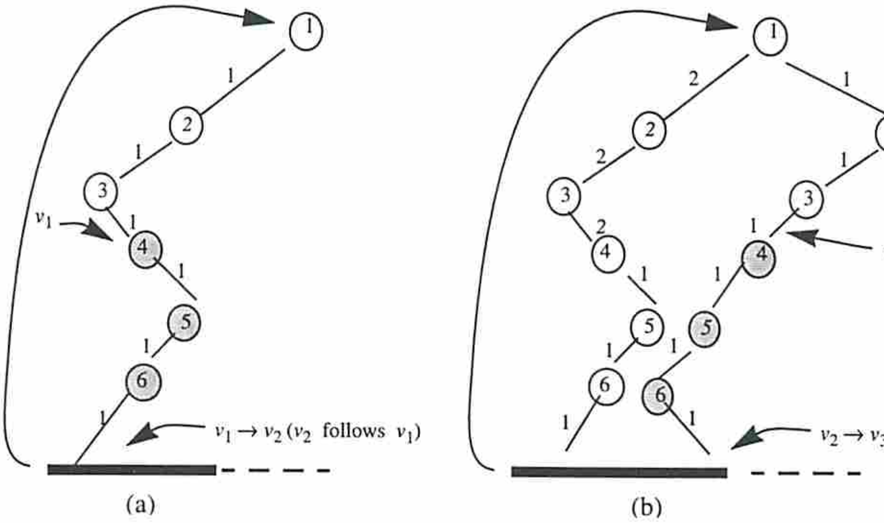


Fig.5

What is important to note here, is that *all* vectors in the original sequence are processed that is, *none of them is skipped* during the construction of DMT_1. This is the theoretical basis for accurate modeling of the input sequences as Markov sources of information. Similarly, continuing this process for all leaves, we end up by building the whole tree DMT_1 as shown in Fig.6.

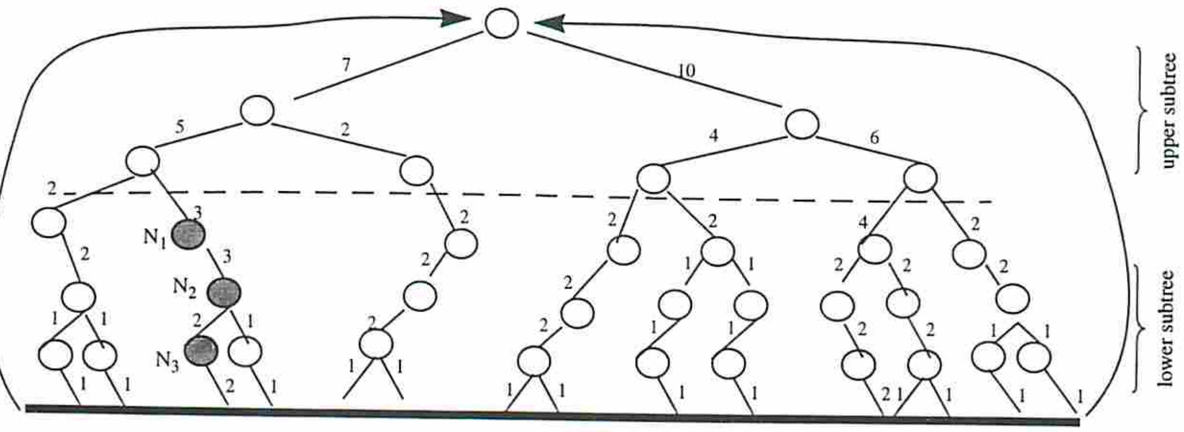


Fig.6

In Fig.6, we separated by a dashed line the two subtrees that constitute DMT_1. The upper subtree (levels 1 to 3) represents DMT_0, that is, it sets up the state probabilities for the sequence; the lower subtrees (levels 4 to 6), give the actual sequencing between any two successive vectors. To keep the counts in these subtrees consistent, while we traverse the lower subtrees and update the counts on their edges, we also accordingly increment the counts on the paths in the upper subtree (in fact, all vectors except the first and the last are processed exactly twice, once in the upper DMT_0 and next in the lower DMT_0).

Obviously, DMT_1 provides more information than DMT_0. To give an example, string '1001' can follow only after '0001' or '1100', information that cannot be gathered by analyzing DMT_0 alone. In addition, the sparse structure of DMT_1 is possible only by using the *dynamic (adaptive)* fashion of growing the tree DMT_0 just illustrated. Another approach would have been to consider as starting point a static binary tree capable to model any 4-bit sequence, to build after that the lower subtree, and finally, to update the counts on the edges while scanning the original sequence. By doing so, we would end up with the obvious disadvantage of having 63 instead of 35 nodes in the structure for the same amount of information; this reason alone is sufficient for considering only dynamically grown models.

The structure DMT_1 just introduced is general enough to capture completely the correlations among all bits of the same input vector and also between successive input patterns. Indeed, the recursive construction of DMT_1 by considering successive bits in the upper and lower subtrees completely captures the word-level (spatial) correlations for each individual input vector in the original sequence. Furthermore, cascading lower subtrees for each path in the upper subtree, gives the actual sequencing (temporal correlation) between successive input patterns. This model captures completely spatial correlations and first-order temporal correlations. 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 (as a function of 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.7.

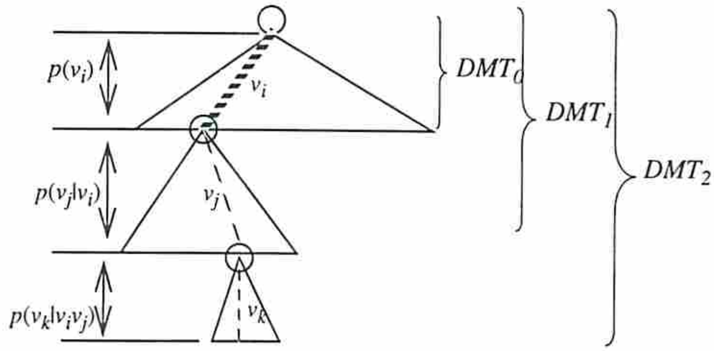


Fig.7

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

Sketch of proof: Let $v = v_1\ v_2 ...\ v_n$ be a string in DMT_k (the substring v_i belongs to the i-level tree). Using Proposition 2, we have $P\ (v_k \mid v_1\ v_2 ...\ v_{k\text{-}1}) = P\ (v_1\ v_2 ...\ v_k)\ /\ P\ (v_1\ v_2 ...\ v_{k\text{-}1})$ and thus the lag-k Markov chain characterizing the input can be fully modeled by the DMT_k structure. ■

3.3 A DMC-based vector compaction procedure[1]

During a one-pass traversal of the original sequence (when we extract the bit-level statistics of each individual vector $v_1, v_2 ..., v_n$ and also those statistics that correspond to 2 consecutive vectors (v_1v_2), (v_2v_3), (v_3v_4),..., $(v_{n\text{-}2}v_{n\text{-}1})$, $(v_{n\text{-}1}v_n)$) we grow simultaneously the tree DMT_1. We continue to grow DMT_1 up to the end of the original sequence. Each generation phase is driven by the user-specified compaction parameter *ratio* that is, we generate a total of $m = n/$ *ratio* vectors. We also note that this strategy does note allow 'forbidden' vectors that is, those combinations that did not occur in the original sequence, will not appear in the final compacted sequence either. This is an essential capability needed to avoid 'hang-up' ('forbidden') states of the sequential circuit during simulation process for power estimation.

Example 5: Coming back to Example 4, let's assume that our objective is to compact this sequence with a compaction ratio of 2. Once we get the Markov tree in Fig.6, we start the generation procedure with parameter *ratio* = 2 and generate a subset of 8 vectors which best approximate the original sequence. To this effect, we use a modified version of the *dynamic weighted selection algorithm* [20]. In that approach, a similar structure with DMT_0 is built; more precisely, a full tree having on the leaves the symbols that need to be generated. The counts on the edges are dynamically updated and the symbols are generated according to their probability distribution. For this, a single random number generator is required in order to divide the interval [0,1] into subintervals that correspond to symbols' probabilities. At each level, the random number is compared to the left probability: if lower, a zero value is generated; if greater, a one value is generated and the number is decreased by the left probability. In our case, we also have to parse simultaneously the upper tree from the root to the leaves, according to the bits generated in the lower subtree. The procedure is then resumed until the needed number of vectors is generated.

In our example, if we assume that $x = 0.23$ is the first randomly generated number, based on the tree in Fig.6, since $0.23 < 7/17$ we take the left edge, generate a value 0 and x remains unchanged. At the second level, $x = 0.23 < 5/$

1. For the sake of simplicity, in this section, we will give the details of the procedure for sequence generation based on DMT_1.

17 so again we generate a '0' and leave x unchanged. Now $x = 0.23 > 2/17$ so a '1' is generated and x becomes $x = 0.23 - 2/17 = 0.11$. For the lower subtree rooted at the node denoted by the vector '001' (that is, we parse the upper subtree according to the already generated bits 0, 0, 1), to produce the second vector, we again generate a random number, say 0.65. At node N_1 in Fig.6, the only choice is to take the right edge, generating a '1'. Next, at node N_2, since $x = 0.65 < 2/3 = 0.66$[1] we take the left edge (generating a '0') and x remains unchanged. At the last level, at node N_3, the decision is quite simple as we have only one descendent. Thus, after the first vector '001', we generate '101' as the second vector. The generation procedure continues for the lower subtree rooted at the node denoted by the vector '101' until the desired length $m = n/ratio$ is achieved. The discussion for DMT_1, captures the essence of the general case involving DMT_k. The procedure to construct DMT_k and generate the compacted sequence is a natural extension of the procedure just discussed.

4. Practical considerations

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.

Input sequences having a large number of bits b are very common in practice; the success of DMC models for sequence compaction when k is large is based on two key observations:

- The larger the value of k is, the sparser the structure of DMT_k will be.

To motivate this, assume a finite input sequence of length n ($n \ll 2^b$). Intuitively, in a worst-case scenario when DMT_k is completely skewed (that is, all vectors are distinct), DMT_k will have a number of nodes proportional to $(k+1)nb$ (in all other cases, due to the sharing of paths among nondistinct vectors, the number of nodes will be smaller). On the other hand, the corresponding full tree (statically constructed) with the same depth, will have a number of nodes proportional with $2^{(k+1)b}$. Therefore the sparsity of the tree DMT_k (compared to the corresponding full tree) will increase with k as: $Sparsity \propto \frac{(k+1)nb}{2^{(k+1)b}}$. Assuming for instance an input sequence on 60 bits having a length of 100,000 vectors, then the sparsity of DMT_1 is about 10^{-29}. The DMC modeling technique exploits this observation by starting with an initially empty model and dynamically growing the Markov tree that characterizes the input sequence. By doing so, one can expect to build much smaller trees than the ones otherwise obtained by using a static model based on an initial full tree. Indeed, in practice the dynamic growing of the Markov model performs very well and the experimental results presented in the next section will support this claim.

- Biased sequences which usually occurs in practice as candidates for power estimation, contain a relatively small number of distinct patterns which arise in many different contexts in the whole sequence therefore a probabilistic model is ideally suited for modeling them.

We point out that both these observations can be efficiently exploited only by a probabilistic technique such as DMC modeling; a deterministic technique (e.g. [10]) has no such inherent capability and therefore cannot avoid all the difficulties that arise from this type of complexity.

---

1. Note that in the lower trees we use conditional probabilities to generate the next vector

5. Experimental results

The overall strategy is depicted in Fig.10.

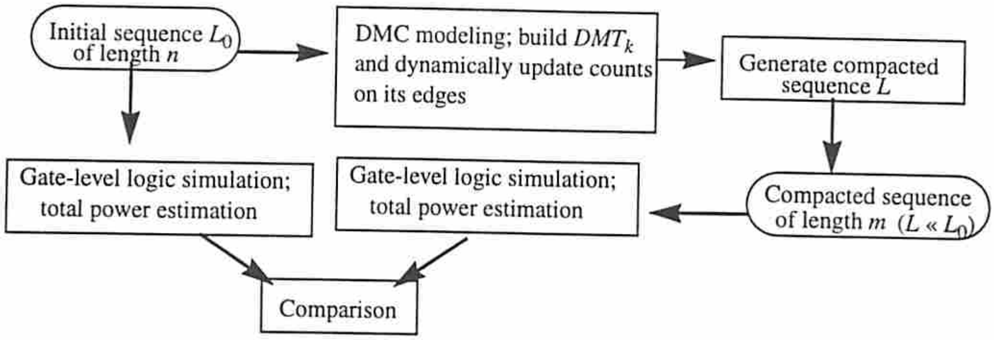


Fig.8

Basically, we verified our ability to compact large input sequences which may also be used as power benchmarks in the design process. We assume that the input data is given in the form of a sequence of binary vectors.

Starting with an k-bit input sequence of length n, 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.

The next step in Fig.8 does the actual generation of the output sequence (of length m). As explained in Section 3, to generate the new sequence we use a modified version of the dynamic weighted selection algorithm presented in [20]. If the initial sequence has the length n and the new generated sequence has the length $m < n$ 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 $r = n/m$ was achieved.

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 *mcnc'91* 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 1, we provide only the real-delay power dissipation results for different initial sequences having the total length of 4,000 vectors; these sequences were produced using a second order information source based on Fibbonacci series. As shown in Table 1, the sequences were compacted with three different compaction ratios (namely r = 2, 5 and 10) using two Markov models: one of order one (that is based on DMT_1) and another one having order two. We give in this table the total power dissipation measured for the initial sequence (column 3) and for the compacted sequence using both models (columns 4-9). On a Sparc 20 workstation with 64 Mbytes of memory, the time necessary to read and compress data was less than 3 sec. for both models. Since the compaction with DMC modeling is linear in the number of nodes in the structure DMT_k, these time values are far less than the actual time needed to simulate the whole sequence. During these experiments, the number of nodes allowed in the Markov model was 10,000 on average.

Table 1: Total Power (uW@20MHz) for sequences of order 2

Circuit	Number of inputs/FFs	Power for initial seq.	Power for $r = 2$ Order 1	Power for $r = 2$ Order 2	Power for $r = 5$ Order 1	Power for $r = 5$ Order 2	Power for $r = 10$ Order 1	Power for $r = 10$ Order 2
bbara	4/4	747.10	826.02	747.69	838.12	748.22	866.76	744.99
bbsse	7/4	1670.20	1781.30	1674.50	1815.90	1681.50	1856.20	1640.20
bbtas	2/3	337.83	364.47	336.75	373.40	335.30	385.50	333.77
cse	7/4	2570.51	2381.60	2567.60	2212.90	2540.90	2174.20	2524.90
dk17	2/3	1439.43	1289.70	1439.80	1281.30	1438.10	1250.20	1438.00
donfile	2/5	3020.31	2836.60	3017.90	2768.80	3015.10	2678.50	3009.30
ex1	9/5	3159.29	2910.10	3100.90	2786.70	3051.30	2575.70	3013.40
ex3	2/4	1542.74	1371.20	1540.80	1348.50	1538.50	1342.70	1534.80
ex4	6/4	1393.86	1610.90	1402.10	1644.00	1414.30	1684.40	1425.70
ex7	2/4	1103.17	1190.00	1105.40	1200.40	1107.00	1225.60	1111.90
mark1	5/4	1344.53	1255.10	1341.10	1189.90	1340.20	1537.70	1338.80
mc	3/2	295.84	241.69	293.98	212.39	291.11	196.76	287.85
opus	5/4	1481.70	1318.20	1477.90	1252.10	1475.20	1124.50	1472.20
planet	7/6	8517.14	5635.50	8299.10	4649.90	8046.20	3596.80	7565.50
sand	11/5	5682.09	5109.70	5619.30	4822.50	5584.70	4416.60	5344.40
shiftreg	1/3	144.60	117.21	144.32	115.26	144.22	109.73	143.84
		Avg. % err.	11.89	0.54	15.74	1.15	20.04	2.20

As we can see, for the model of order 2, the quality of results is very good even when the length of the initial sequence is reduced by one order of magnitude. Thus, for *ex4* in Table 1, instead of simulating 4,000 vectors with an exact power of 1393.86 uW, one can use only 800 vectors ($r = 5$) with an estimate of 1414.30 uW or just 400 vectors ($r = 10$) with power consumption estimated as 1425.70 uW. 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. On the other side, using a first-order model, the quality of the results can be seriously impaired. For instance, in the case of benchmark *planet*, if $r = 2$ we can erroneously predict a total power of 5635.50 (33.83% relative error) or a value of 3596.80 (57.78% error) if $r = 10$. This is because for a sequence generated with a second-order source, a model that considers only pairs of two consecutive vectors cannot preserve correctly even the first-order transition probabilities for the primary inputs and state lines ($p(x_n s_n x_{n-1} s_{n-1})$ in our notation).

Next, we studied the sensitivity of the proposed approach to the choice of initial seeds used for random excitation of the DMC model. Using different seeds for the random number generator (and therefore choosing different initial states in the sequence generation phase), we run a set of 1,000 experiments for the DMC technique. We report in Table 2 the percentage of error violations for total power values, using as thresholds 5%, 6% and 10%.

Table 2: Results obtained for 1,000 runs

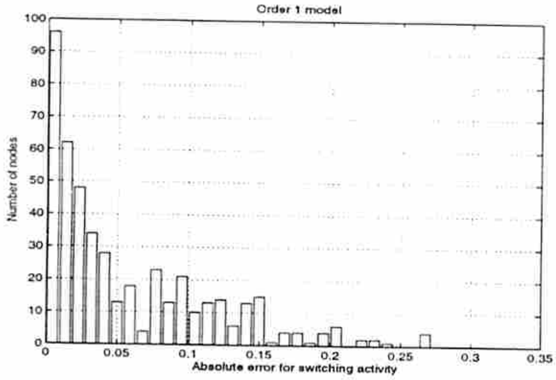
Circuit	Number of vectors	Error violations for order 1 model (%) > 5%	> 6%	>10%	Error violations for order 1 model (%) > 5%	> 6%	>10%
bbara	2000	25.2	9.9	0.1	0.0	0.0	0.0
bbsse	400	14.3	8.3	0.1	0.0	0.0	0.0
bbtas	800	8.9	4.5	0.1	0.0	0.0	0.0
cse	800	13.4	7.7	0.4	0.0	0.0	0.0
dk17	400	98.8	91.4	4.7	0.0	0.0	0.0
donfile	400	12.8	6.2	0.3	0.0	0.0	0.0
ex1	800	13.8	7.7	0.2	0.0	0.0	0.0
ex3	2000	100.0	100.0	3.1	0.0	0.0	0.0
ex4	400	79.0	72.3	38.8	0.0	0.0	0.0
ex7	2000	11.1	2.3	0.0	0.0	0.0	0.0
mark1	800	10.5	6.1	0.2	0.0	0.0	0.0
mc	2000	89.5	81.5	35.4	0.0	0.0	0.0
opus	2000	23.6	12.0	0.2	0.0	0.0	0.0
planet	800	98.1	97.6	89.4	1.2	0.0	0.0
sand	800	21.6	14.1	1.6	0.0	0.0	0.0
shiftreg	400	99.9	99.8	87.7	0.0	0.0	0.0

Once again, the results obtained with the second-order DMC score very well over the first-order model and prove the robustness of the present approach. As we can see in Table 2, the accuracy is higher in all cases when an appropriate Markov model is used. The percentage of error violations was 0 (except for circuit *planet*) for the second-order model, while the first-order one is far behind.

To assess the importance of correctly modeling the input sequence, we give in Table 3 our results for configurations b) and c) in Fig.3 with a compaction ratio of 5. In the first case we cascaded benchmarks *ex4* (from *mcnc91* suite) and *s1196* (from *ISCAS'89* suite) and we estimated the total power consumption for both of them. In the second case, we used a complex topology where benchmarks *ex3* and *planet* interact. Looking at the results in Table 3 we can conclude that only the second order model is appropriate for this type of analysis.

Table 3: Total Power (uW@20MHz) for sequences of order 2 for different configurations

Configuration	Number of inputs/FFs	Power for initial seq.	Power for order 1	Power for order 2
cascade (b)	6/22	5762.03	6158.38	5772.68
interacting (c)	6/10	11278.65	10290.09	11188.82
		Avg. % err.	7.82	0.49



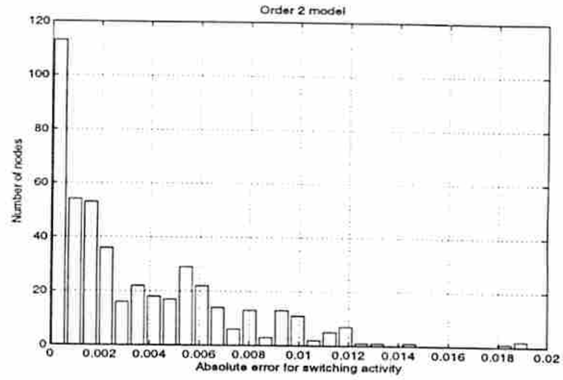


Fig.9

Finally, we give in Fig.9 a *node-by-node analysis* of switching activity for benchmark *planet* for a compaction ratio of 5. As we can see, using a lower order model than needed may significantly impair the ability of correctly estimating the switching activity on a node-by-node basis. While under the assumption of a first order model the absolute error (defined as $|sw_{comp}-sw_{exact}|$ where sw_{comp} is the switching activity obtained using the compacted sequence for simulation) 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. We found this kind of behavior typical for the whole set of experiments we performed.

7. Conclusion

In this paper we investigated from a probabilistic point of view the effect of finite-order statistics of the input sequence on FSM and interacting FSM behavior. Based on dynamic Markov modeling, we proposed an effective approach to compress an initial sequence into a much shorter one such that the steady state and transition probabilities (and therefore the total power consumption) in the target circuit are preserved.

The mathematical foundation of this approach relies on adaptive modeling of binary input streams as first- and higher-order Markov sources of information. For the first time to our knowledge, the effect of temporal correlations longer that one clock-cycle on the power dissipation in FSMs and networks of interacting FSMs was studied. As shown by the experimental results, large compaction ratios can be obtained with less than 3% loss in accuracy for total and node-by-node power consumption.

The results presented in this paper represent an important step towards understanding the FSM behavior from a probabilistic point of view.

References

[1] K. Parker and E.J. McCluskey, 'Probabilistic Treatment of General Combinational Networks', in *IEEE Trans. on Computers*, vol. C-24, June 1975.

[2] J. Jain, J. A. Abraham, J. Bitner, and D. Fussell, 'Probabilistic verification of Boolean Functions', in *Formal Methods in System Design*, vol.1, pp. 61-115, 1992.

[3] J. Savir, G.S. Ditlow, and P.H. Bardell, 'Random Pattern Testability', in *IEEE Trans. on Computers*, vol. C-33, Jan. 1984.

[4] P. H. Bardell, W. H. McAnney, and J. Savir, 'Built-in Test for VLSI: Pseudorandom Techniques', J. Wiley & Sons Inc. 1987.

[5] J. Rabaey and M.Pedram, in 'Low-Power Design Methodologies', Kluwer Academic Publishers, 1996.

[6] A. Papoulis, 'Probability, Random Variables, and Stochastic Processes', McGraw-Hill Co., 1984.

[7] G. Hachtel, E. Macii, A. Pardo, and F. Somenzi, 'Probabilistic Analysis of Large Finite State Machines', in *Proc. ACM/IEEE Design Automation Conference*, pp. 270-275, June 1994.

[8] C.-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M. Despain, and B. Lin, 'Power Estimation Methods for Sequential Logic Circuits', in *IEEE Trans. on VLSI Systems*, vol.3, no.3, Sept. 1995.

[9] M. Pedram, 'Power Minimization in IC Design: Principles and Applications', in *ACM Transactions on Design Automation of Electronic Systems,* vol.1, no.1, pp.1-54, Jan.1996.

[10] J. Monteiro and S. Devadas, 'Techniques for Power Estimation of Sequential Logic Circuits Under User-Specified Input Sequences and Programs', in *Proc. Intl. Workshop on Low Power Design,* pp. 33-38, April 1994.

[11] D. Marculescu, R. Marculescu, and M. Pedram, 'Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation', in *Proc. ACM/IEEE Design Automation Conference*, pp. 696-701, June 1996.

[12] C.-Y. Tsui, R. Marculescu, D. Marculescu, and M. Pedram, 'Improving the Efficiency of Power Simulators by Input Vector Compaction', in *Proc. ACM/IEEE Design Automation Conference*, pp. 165-168, June 1996.

[13] T. Bell, J. Cleary, and I. Witten, 'Text Compression', Prentice Hall, 1990.

[14] G.V. Cormack and R.N. Horspool, 'Data Compression Using Dynamic Markov Modeling', in *Computer Journal,* Vol. 30, No. 6, pp. 541-550, 1987.

[15] S. P. Meyn, and R. L. Tweedie, 'Markov Chains and Stochastic Stability', Springer-Verlag, 1993.

[16] T.M. Cover, and J.A. Thomas, 'Elements of Information Theory', John Wiley & Sons, Inc., 1991.

[17] S. Devadas and A.R. Newton, 'Decomposition and factorization of Sequential Finite State Machines', in *IEEE Trans. on Computer*-Aided Design of Integrated Circuits, vol.8, No.11, pp. 1206-1217, Nov. 1989.

[18] C.J. Burke and M. Rosenblatt, 'A Markovian Function of A Markov Chain', in *Ann. Math. Statist.*, vol.29, pp. 1112-1122, 1958.

[19] T.E. Harris, 'On Chains of Infinite Order', in *Pacific J. Math.*, vol. 5, pp. 707-724, 1955.

[20] J.W.Green and K.J.Supowit, 'Simulated Annealing without Rejected Moves', in *Digest. of Intl. Conference on Computer Design,* pp. 658-663, Oct. 1984.