Hierarchical Sequence Compaction for Power Estimation*

Radu Marculescu, Diana Marculescu, Massoud Pedram

Department of Electrical Engineering - Systems
University of Southern California, Los Angeles, CA 90089

Abstract - This paper presents an effective technique for compacting a large sequence of input vectors into a much smaller one such that when the two sequences are applied to any circuit, the resulting power dissipations are nearly the same. 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. The new framework has a high degree of adaptability, i.e. the hierarchical model is dynamically grown according to the sequence behavior. Experimental results demonstrate that large compaction ratios can be obtained without significant loss in accuracy (less than 5% on average) for power estimates.

I. Introduction

With the growing need for low-power systems, power analysis and low-power synthesis have become primary concerns for the design community. Power estimation is in general a difficult problem. To date, both simulative [1][2] and nonsimulative approaches [3]-[6] have been tried, each one having its own advantages and limitations. More specifically, general simulation techniques provide sufficient accuracy, but have high computational cost. On the other hand, nonsimulative approaches are in general faster, but are less accurate.

Referring to simulation based techniques, the *input statistics* and the *length of the input sequences* are two important issues for power estimation. Our objective in this paper is to solve efficiently the following problem: having an initial sequence (assumed representative for the application data), transform that input sequence into a smaller one, such that the new body of data represents a *good approximation* of the initial sequence as far as total power consumption is concerned.

Attempts to solve this probleme do exist. Elaborate and effective techniques were presented in [7][8] where authors succeed in compacting large sequences without significant loss in accuracy. However, both techniques may introduce new vectors in the final compacted sequence and do not adapt very well to changes in the input characteristics. To illustrate the significance of this latter issue, we consider an example.

Example 1: Let S_1 be a 5-bit sequence as shown in Fig.1a; next to it, we represent the word-level transition graph that corresponds to this sequence. Each state in this graph corresponds to a distinct pattern in the sequence and each edge represents a valid transition between any two patterns that occur in the sequence; the label of each edge captures the conditional probability of transition from the source node to the destination node. This particular set of inputs behaves like a pseudorandom sequence because any vector is equally likely to be followed by any other remaining pattern. The total number of bit-flippings in the whole sequence is 36; then, dividing this value by the sequence length, we get an average value of 3 transitions per time step.

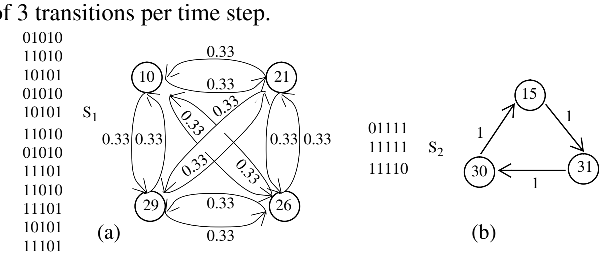

Fig.1: Two sequences and their corresponding transition graphs

In Fig.1b 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 S_1* and S_2*, respectively. Based on S_1* and S_2*, we construct now a new sequence S* which is formed by concatenating S_1* and S_2* in the order $S_1^* \rightarrow S_2^* \rightarrow S_1^*$; the transition graph representation of this new 'macrosequence' is given in Fig.2.

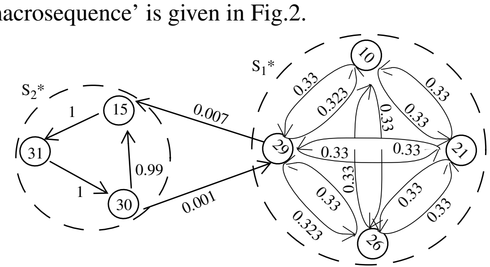

Fig.2: The transition graph for the composite sequence S^*

The question becomes now, how will the average power consumption look like as a function of time, when S* is applied to any circuit? Obviously, the circuit has now two very different modes of operation: one where a lot of activity is generated at the primary inputs, and a second one where about one single input bit toggles at every time step. In Fig.3 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 110µW; after that, when the characteristics of the input sequence change, the power value goes down toward 70µW and finally, due to the increase of the switching activity at the primary inputs, it comes up towards 90µW.

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 stimuli may contain a mixture of vectors, each one very different as far as average switching activity per bit is concerned.

*This research was supported by DARPA under contract F33615-95-C1627, SRC under contract 97-DJ-559, NSF under contract MIP-9457392 and a grant from Toshiba Corp.

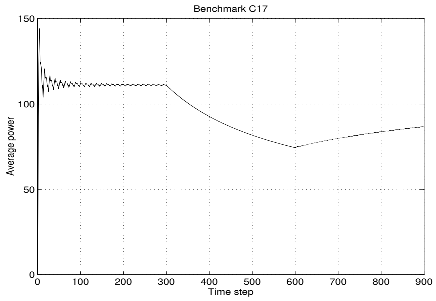

Fig.3: Average power dissipation for C17

The approach suggested in [7] tries to model the *average behavior* which might result when such a mixture of vectors is present at the primary inputs of the circuit. To this end, the authors synthesize first a stochastic machine that is probabilistically equivalent to the initial sequence. After that, by applying randomly generated inputs to the stochastic machine, a random walk through the states of the transition graph is emulated and a new and shorter sequence is generated. However, 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. As a consequence, an erroneous power value can be obtained depending on which component (low or high activity) is visited. The same type of phenomenon is observed for statistical methods when the distribution is very different from a normal one (e.g. bi-modal distributions), or when one selects from the initial sequence only the first few hundred vectors. Thus, in practice, to compact large sequences that contain non-homogeneous power behaviors, a technique with high adaptability is needed.

The present paper improves the-state-of the art by providing an original solution for the vector compaction problem. The foundation of our approach is probabilistic and relies on *adaptive (dynamic) modeling* of binary input streams as first-order Markov sources of information. As distinctive feature, we use hierarchical Markov models to structure the input space into a hierarchy of macro- and micro-states: 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. Another important property of such models is to capture the different operating modes of a circuit using the first level in the hierarchical Markov model, thus providing high adaptability to different operating modes of the circuit.

After constructing the hierarchy for an 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. Next, the control 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. While the compaction procedure in [7] can be adapted to work in this new environment, we prefer instead to combine the advantages offered by the hierarchical model with a new and computationally more efficient technique, called dynamic Markov chain (DMC) modeling. The initial formulation of DMC modeling [9], was extended in [10] to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive input patterns.

This new framework is very effective in power estimation. The basic idea is illustrated in Fig.4. To evaluate the total power consumption of a target circuit for a given input sequence L_0 (Fig.4a), we derive first the hierarchical Markov model (HMM) of the input sequence and after that, 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.4b).

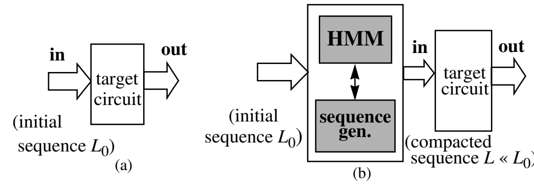

Fig.4: Sequence compaction using HMM

The paper is organized as follows: Section II formalizes the power-oriented vector compaction problem and discusses parameters that make this approach effective in practice. Section III introduces and characterizes the concept of hierarchy associated with the input space. Section IV presents a DMC-based procedure for vector compaction and finally, in section V, we give some practical considerations and experimental results. We conclude by summarizing our main contribution.

II. Power Oriented Data Compaction

Capturing only signal probabilities at the primary inputs of the circuit is not enough for accurate power estimation therefore it is critical to distinguish between sequences which exhibit the same signal probabilities on different bit lines, yet showing very different spatial and temporal correlations. 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: $P_{avg} = \left(\frac{f_{clk}}{2}\right) \cdot V_{DD}^2 \cdot \sum_n (C_n \cdot sw_n)$, 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. Hence, the average switching activity per node (gate) is the key parameter that needs to be correctly determined, especially if we are interested in a node-by-node power estimation. Consequently, the vector compaction problem can be formulated as follows: for any k-bit sequence of length n (consisting of vectors $v_1, v_2, ..., v_n$), find another sequence of length $m < n$ (consisting of the subset $u_1, u_2, ..., u_m$ of the initial sequence), such that the average transition probability on the primary inputs is preserved *wordwise.* More formally, for any generic input v and u (seen as a collection of bits) in the original and in the compacted sequence, respectively, the following holds:

$$\left|p(v^- = \alpha \wedge v^+ = \beta) - p(u^- = \alpha \wedge u^+ = \beta)\right| < \varepsilon \qquad (1)$$

In relation (1), v^-, v^+ (u^-, u^+) denote the current and the next vector, respectively, in the original (compacted) sequence and α, β are any two patterns that appear in the initial sequence. The condition above simply requires that the joint transition probability for any group of bits is preserved within a given level of error, for any two consecutive vectors.

III. Hierarchical Modeling of The Input Sequence

Our objective now is to structure the transition graph associated with an input sequence into a hierarchy of subgraphs that correspond to different behaviors (in particular, different power consumptions). To this end, we provide first some useful definitions.

A. Micro/Macro-state modeling

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

$$p(v_n|v_{n-1}v_{n-2}\ldots v_1) = p(v_n|v_{n-1}) \qquad (2)$$

where $p(v_n|v_{n-1}) = p(v_n v_{n-1})/p(v_{n-1})$ is the conditional probability of v_n given v_{n-1}.

In other words, the probability of having a symbol depends only on the previous vector and is independent on the 'history' of the sequence.

Definition 2. (Weighted transition graph). A *weighted* transition graph is a directed graph where any edge from state s_i to state s_j is labelled with a conditional probability $p(s_j|s_i)$ *and* a weight w_{ij} associated with the transition $s_i \rightarrow s_j$.

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

Definition 3. (Weight of a random walk). The weight of a random walk in a weighted transition graph is given by:

$$W = \sum_{s_i,\, s_j} p(s_i) \cdot p(s_j|s_i) \cdot w_{ij}$$

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

Definition 4. ((ε, δ)-property). A weighted transition graph is said to have the (ε, δ)-property if there exists a grouping $\{S_1, S_2,...,S_p\}$ on the set of states $\{s_1, s_2,..., s_n\}$ 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): $\forall\ S_k,\ \exists\ W_k$ such that $|\ W_k - w_{ij}\ | < \delta$, for any two states $s_i, s_j \in S_k$. Also, if $k < l$, W_k's are such that $W_k < W_l$. S_k's are called the *macrostates* whereas $s_i \in S_k$ are called the microstates within macrostate S_k.

The intuitive reason for the above definition is the following: 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.2 microstates '10', '21', '26', '29' form the macrostate S_1* (with high activity), while '15', '30', '31' form S_2* (with low activity).

In general, a particular microstate may 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 4, the weight value for a transition determines whether the microstates belong to the macrostate or not). Therefore, the *grouping* of states is done such that transitions are *clustered* according to their associated weights.

From what we defined so far, we are able to structure the input space hierarchically. More precisely, instead of considering the input sequence as a flat sequence of vectors we can see it as a structured multi-level discrete stochastic process called *Hierarchical Markov Model* (HMM). 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 HMM as well. For instance, the graph in Fig.2 can be represented hierarchically as shown below, where the macrostate S_1* identifies the high activity mode and S_2* the low activity one.

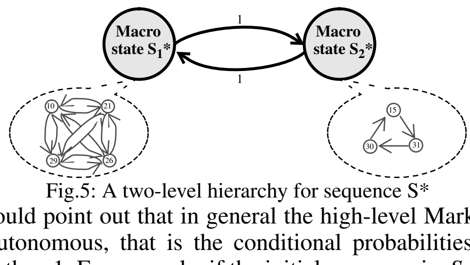

Fig.5: A two-level hierarchy for sequence S*

We should point out that in general the high-level Markov chain is not autonomous, that is the conditional probabilities may be different than 1. For example, if the initial sequence is: $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1$ (thus having three modes of operation), then in the high-level Markov chain we have $p(S_3|S_2) = p(S_1|S_2) = 0.5$ (because from S_2 is equally likely to go to either S_3 or S_1).

The initial problem of compacting a flat input sequence becomes now that of compacting a hierarchy of subsequences. Since the vectors from each macrostate are gathered using the same δ–criterion, the compaction is done now inside each macrostate. This way, the 'hang-up' problem mentioned in the introductory part is avoided, that is all macrostates are guaranteed to be visited (as their transition probabilities 'scale-up' after hierarchization). For instance, in Fig.2, the transition probabilities between S_1* and S_2* equal 0.001 and 0.007 respectively; in the hierarchical organization shown in Fig.5, these probabilities become 1.

In what follows we present some useful results for HMM characterization. (Proofs are given in [14].)

Theorem 1. If the state probability of each macrostate and the state probabilities for all microstates within a macrostate are preserved, then the state probability distribution for the initial (flat) sequence is completely captured.

Theorem 2. In a hierarchical model satisfying the ε–criterion, if transition probabilities of the microstates are preserved within each macrostate and if the state probabilities of the macrostates are correctly captured, then transition probabilities of the initial sequence are reproduced with an error less than or equal to ε.

Theorem 3. If the hierarchy satisfies the (ε, δ)-property (as in Definition 4), then the weight of a random walk in the flat model satisfies:

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

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 4 and $p(S_k S_l) = p(S_k)\ p(S_l|S_k)$ is the transition probability from macrostate S_k to macrostate S_l.

In other words, a random walk on the HMM *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 among them.

This general formulation applies immediately to our problem defined in Section II. In fact, if the input sequence is hierarchically structured, Theorem 2 guarantees that inequality (1) is still satisfied. Moreover, Theorem 3 guarantees that the average power value is maintained. Practically, this is very important because the hierarchical model has the advantage of being *highly adaptive* as opposed to a flat processing of the input which does well 'on average'.

B. A Hamming distance-based criterion for microstate grouping

In practice, it is hard to determine the weight w_{ij} for each individual transition. This would mean to have detailed information about the circuit (e.g. capacitive loads, internal structure) and to employ a simulation procedure to derive the exact power consumption for each pair of vectors from the original sequence [13]. For all practical purposes, this is at least inconvenient if not impossible, and therefore we suggest a different, *circuit independent*, criterion to structure the input space. As suggested in Example 1, what we need is an indicator for the level of activity at the primary inputs. For this purpose, we use the *average Hamming distance* between two consecutive vectors because, from our investigation, it seems to be a reliable indicator of the average power consumption. However, the framework we provide is open to other, more elaborate, weighting functions.

In this particular example (see Fig.6), based on the Hamming distance criterion, we can roughly classify the input sequence into 'high activity' and 'low activity' macrostates (that is, if more than

3 out of 5 bits change, then we are in the high activity mode, otherwise in low activity mode). While this kind of partitioning into high and low activity modes can be always used, in practice it is better to have a more refined model.

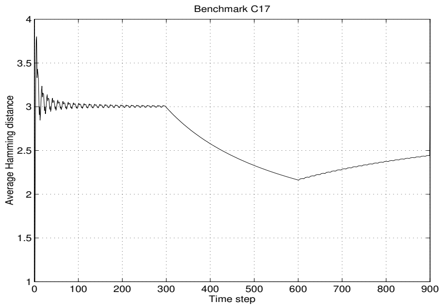

Fig.6: Average Hamming distance variation

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 if we are not only interested in preserving the total power consumption, but are also required to identify the different modes of operation (e.g. an initialization mode, a normal operation mode, a standby mode, and a sleep mode). To detect the changes in the input sequence, a *sliding window* is used to compute the average value of the predictor function. We note that the *size* of the window should not be chosen too small (due to the fragmentation, the high-level Markov chain becomes very similar to the flat model) or too large (the low-level Markov chain becomes very similar to the flat model). However, our experience shows that a window size of 50-100 vectors works well in practice. We should note that the ε-criterion (if satisfied) is already taken care of by this procedure: since all macrostates are guaranteed to be visited (due to the scaling of conditional probabilities in the high-level model), we cannot end up with a wrong value for the total power consumption as is the case for the flat model.

IV. A DMC-Based Vector Compaction Procedure

A. Background on dynamic Markov models

Suppose that the set of events of interest is the set S of all finite binary sequences on k 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. Imposing a total ordering among bits, such a sequence may be conveniently represented as a binary tree DMT_0 (*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) which indicates how many times the substring from the root to that particular node, occurs in the original sequence.

The construction procedure for trees DMT_0 and their properties are described in detail in [10]. However, we observe that DMT_0 alone cannot capture temporal correlations because the relative order of vectors in the initial sequence is irrelevant for the construction of DMT_0. Consequently, we refine now the above structure by incorporating *first-order temporal effects* and defining a new tree called DMT_1 (*Dynamic Markov Tree of order 1*).

Example 2: For the following 4-bit sequence consisting of 8 non-distinct vectors: (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) = (0000, 0001, 1001, 1100, 1001, 1100, 1001, 1100) the construction of the tree DMT_1 is shown in Fig.7. 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_8 is considered as being bit number one (and consequently represented at level one in DMT_1), the next bit is considered as being bit number two and so on. Every time a vector is completely scanned (this corresponds to reaching level four in the tree), we construct a new tree, rooted at that node and representing the next vector. For instance, vector v_2 = 0001 follows immediately after v_1 = 0000; consequently, when we reach the node that corresponds to v_1 (the leftmost path in Fig.7a), instead of going back to the root (and therefore 'forgetting' the context), we start to build a new tree, rooted at the current leaf. The newly constructed tree will preserve the context in which v_2 = 0001 occurred that is, immediately after v_1 = 0000 (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.7b.

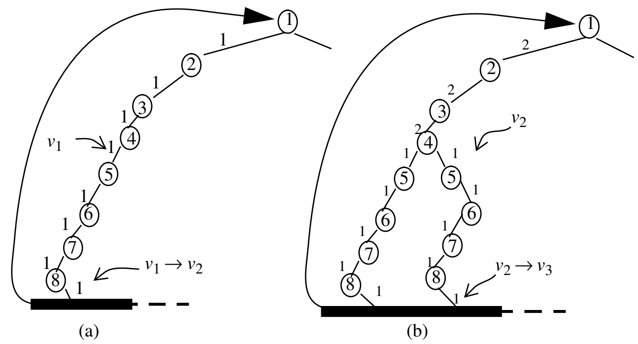

Fig.7: Construction of first-order dynamic Markov trees

What is important to note here, is that *all* vector pairs 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 first-order Markov sources of information. Continuing this process for all vectors we end up by building the whole tree DMT_1 as shown in Fig.8. The upper subtree in DMT_1 (levels 1 to 4) represents in fact DMT_0, that is, it sets up the state probabilities for the sequence; the lower subtrees (levels 5 to 8), 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. Obviously, DMT_1 provides more information than DMT_0. To give an example, from Fig. 8, we can see that string '1001' can follow only after '0001' or '1100', information that cannot be gathered by analyzing DMT_0 alone.

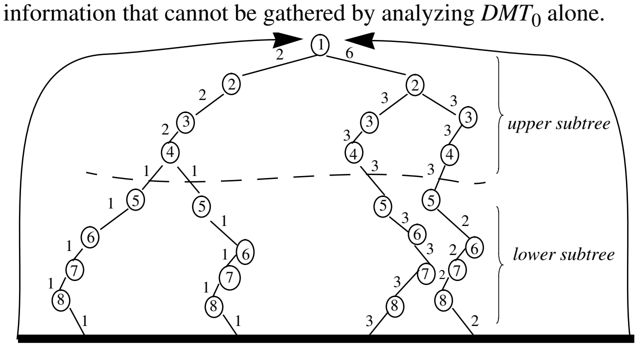

Fig.8: Structure of DMT_1

Proposition 1. At every node in DMT_0 we have:

$$p(v) = \sum_{x \in S} p(vx) \tag{3}$$

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

The above condition, simply states that the sum of the counts attached to the immediate successors of node v equals its own value $p(v)$. As we can easily see in Fig.8, condition (3) is satisfied at every node in this representation. 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 '1001' is 3/8 (because the count on the terminal edge that corresponds to '1001' is 3 and the length of the sequence is 8) while the probability of string '1111' is zero, '1111' being a 'forbidden' vector for this particular sequence.

Proposition 3 [11]. We write the probability of a vector string $v = v_1 v_2 ... v_n$ as follows:

$$P(v) = P(v_1) \cdot P(v_2 | v_1) \cdot \ldots \cdot P(v_n | v_1 v_2 \ldots v_{n-1}) \quad (4)$$

This property, used in connection with the counts on the edges, allows a quick calculation of the transitions probabilities that characterize a particular sequence. For example, if we want to calculate the transition probability '1001' → '1100' we have from Proposition 3 $P(v) = P(v_1 v_2) = P(v_1) \cdot P(v_2 | v_1) = 3/8$ which is exactly the count on the path '10011100' in the tree DMT_1 divided by the sequence length.

Theorem 4. Any sequence in S can be modeled as a first-order Markov source using the structure DMT_1 and its parameters. We call this process Dynamic Markov Chain (DMC) modeling.

B. A practical procedure for sequence generation

We describe now a practical procedure to construct DMT_1 and generate the compacted sequence. First, vectors are assigned to macrostates during a one-pass traversal of the input sequence based on average Hamming distance as explained in Section III.

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 $(v_1 v_2)$, $(v_2 v_3)$,...,$(v_{n-2} v_{n-1})$,$(v_{n-1} v_n)$), 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 change the characteristics (average weight and transition probabilities) of the model by much. 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* [12]. The pseudocode for the generation phase and detailed examples involving flushes are given in [14]. In general, by alternating the generation and flush phases in the DMC procedure, the complexity of the model can be effectively handled. To see how the flushing technique affects the accuracy, assume that an input sequence of length n is modeled by the *DMC* approach. Suppose that during the building of the Markov model, flushing occurs after the first n_1 vectors, then after the next n_2 vectors, and so on. If the number of flushes is f, then $n_1 + n_2 + ... + n_f = n$ and the following result holds:

Theorem 5. The error ε in estimating transition probabilities with a model that supports flushing satisfies:

$$\varepsilon = \frac{1}{n} \cdot \sum_{i=1}^{f} n_i \cdot \varepsilon_i \le max(\varepsilon_i) \quad (5)$$

where ε_i, n_i are the error and the number of vectors processed when the sequence between the i-th and (i+1)-th flushes is generated. Differently stated, we do not have to worry about the number of flushes (needed to manage complexity) if the individual Markov models capture accurately the characteristics of the subsequences.

The only remaining issue is to determine how many vectors must be generated inside each macrostate before a transition to another macrostate is performed. In general, if a subsequence of length L_i is assigned to 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 that of the initial one. 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 we need to generate a number of $L_i/(r \cdot p(S_i) \cdot M)$ vectors. Since we do compaction only at the microstate level, the length of the macro-sequence is preserved (the generation procedure stops when M macrostates are obtained).

We also note that this strategy does *not* allow 'forbidden' vectors that is, those combinations that did not occur in the original sequence, will not appear in the final compacted sequence either.

V. Experimental Results

The overall strategy is depicted in Fig.9.

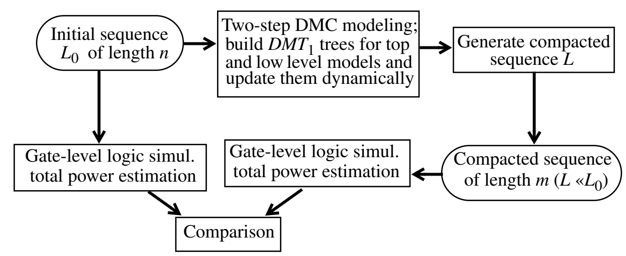

Fig.9: Experimental setup

We assume that the input data is given in the form of a sequence of binary vectors. Starting with a k-bit input sequence of length n, we perform a one-pass traversal of the original sequence to assign microstates to macrostates. Next, we 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 m). If the initial sequence has length n and the new generated sequence has 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 use a real-delay gate-level logic simulator developed under SIS. The total power consumption of some benchmarks has been measured for the initial and the compacted sequences. In Table 1, we provide only the real-delay results for a set of highly biased sequences (of length 4,000) which contains three modes of operation: a high activity sequence duplicated 4 times, followed by a low activity sequence, and finally by a pseudorandom one. As shown in Table 1, the initial sequences were compacted with two different compaction ratios (namely r = 5 and 10); we give in this table the total power dissipation measured for the initial sequence (column 2) and for the compacted sequence (columns 3-6). The time in seconds (on a Sparc 20 with 64 Mbytes of memory) necessary to read and compress data with DMC modeling was below 5 seconds in all cases, for both the flat and the hierarchical models. Since compaction with DMC modeling is linear in the number of nodes

Table 1: Total power consumption for ISCAS'85 circuits

Circ.	Exact power	Flat model $r = 5$	Flat model $r = 10$	Hierarchical model $r = 5$	Hierarchical model $r = 10$
C432	1810.02	1491.58	1230.77	1888.42	1906.84
C499	4390.79	5341.74	6126.58	4497.01	4591.46
C880	3788.22	4504.40	2803.14	3851.42	4006.19
C1355	3783.35	3065.71	4333.81	3910.45	3933.25
C1908	6352.07	4565.87	7094.39	6145.49	6493.44
C3540	14471.46	9005.19	3527.65	15056.43	15021.08
C6288	104158.25	81100.59	82652.47	98112.01	96295.36
	Avg.% err.	23.64	31.45	3.55	4.74

Table 2: Results obtained for SRS and HMM

	Simple Random Sampling					Hierarchical Markov Model			
	# of vector pairs		Error violations (%)			# of vectors	Error violations (%)		
Circ.	Max.	Avg.	> 5%	> 6%	>10%		> 5%	> 6%	>10%
C432	3300	2176	1.1	0.7	0.4	800	2.6	0.6	0.0
C499	1500	862	1.4	1.3	0.2	800	0.1	0.0	0.0
C880	3990	2705	1.8	0.4	0.7	800	2.8	0.9	0.0
C1355	1380	814	1.7	1.0	0.2	800	0.1	0.0	0.0
C1908	1620	846	1.9	1.3	0.2	800	0.1	0.0	0.0
C3540	2340	1446	2.0	1.3	0.4	1200	1.6	0.3	0.0
C6288	7470	5422	1.4	1.4	0.3	3600	1.5	0.0	0.0

in the DMT_1 structure, this value is 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 20,000 on average (around 500Kbytes of memory). The sequences satisfied the ε-criterion for $\varepsilon = 0.001$, while the parameter δ in Definition 4, was set to $0.05 \cdot$(# of input bits).

As we can see, 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 1, instead of simulating 4,000 vectors with an exact power of 1810.02µW, one can use only 800 vectors with an estimate of 1888.42µW or just 400 vectors with a power consumption estimated as 1906.84µW. 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 hand, if one uses the flat model (i.e. a single DMT_1 is built for the whole sequence), for the same benchmark the relative errors in power prediction are 18% and 32%, respectively. The main reason for this inaccuracy is the lack of adaptability which characterizes the flat model when it is applied to multi-modal sequences. This is not true for 'well-behaved' sequences, that is uni-modal sequences, for which the flat model performs very well as reported in [10].

Finally, we compare our results generated by HMM with simple random sampling (SRS) of vector pairs from the original sequences. In simple random sampling, we performed 1,000 simulation runs with 0.99 confidence level and 5% error level on each circuit[1]. We report in Table 2 (columns 2, 3) the maximum and average number of vector pairs needed for total power values to converge. We also indicate the percentage of error violations for total power values, using as thresholds 5%, 6% and 10% (columns 4-6). 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 HMM technique. In Table 2 (columns 8-10), we give the results obtained with the hierarchical model, for the same thresholds as those 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 that of simple random sampling in most of the cases.

VI. Conclusion

In this paper, we addressed the vector compaction problem from a probabilistic point of view. Structuring the input space as a two-level hierarchy, we proposed an original approach to compact an initial sequence into a much shorter equivalent one, which can be used after that 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 sequence characteristics such as different circuit operating modes or varying power distributions. The results obtained on standard benchmarks, show that using this hierarchical model, large compaction ratios can be obtained without much loss in accuracy in total power estimates.

The issues brought into attention represent an important step to reduce the gap between simulative and nonsimulative techniques which are currently the norm.

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

References

[1] F.N. Najm, 'A Monte Carlo Approach for Power Estimation', *IEEE Transactions on VLSI Systems*, Vol.1, No.1, pp. 63-71, Mar.1993.

[2] C.X.Huang, B.Zhang, A.-C.Deng, and B.Swirski, 'The Design and Implementation of PowerMill', in *Proc. Intl. Workshop on Low Power Design,* pp. 105-110, April 1995.

[3] F. N. Najm, 'Transition Density: A New Measure of Activity in Digital Circuits', *IEEE Transactions on CAD*, Vol. 12, No.2, pp. 310-323, Feb.1993.

[4] R. Marculescu, D. Marculescu, and M. Pedram, 'Efficient Power Estimation for Highly Correlated Input Streams', in *Proc. ACM/IEEE Design Automation Conference*, pp. 628-634, June 1995.

[5] D. Marculescu, R. Marculescu, and M. Pedram, 'Information Theoretic Measures for Power Analysis', in *IEEE Trans. on Computer-Aided Design of Integrated Circuits,* vol. 15, No. 6, June 1996.

[6] M. Nemani and F. Najm, 'Towards A High-Level Power Estimation Capability', in *IEEE Trans. on Computer-Aided Design of Integrated Circuits,* vol. 15, No. 6, June 1996.

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

[8] 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.

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

[10] R. Marculescu, D. Marculescu, and M. Pedram, 'Adaptive Models for Input Data Compaction for Power Simulators', in *Proc. Asia and South-Pacific Design Automation Conference*, pp. 391-396, Japan, Jan. 1997.

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

[12] 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

[13] S.-Y. Huang, K.-C. Chen, K.-T. Cheng, and T.-C. Lee, 'Compact Vector Generation for Accurate Power Simulation', *Proc. ACM/IEEE Design Automation Conference,* pp. 161-164, June 1996.

[14] R. Marculescu, D. Marculescu, and M. Pedram, 'Vector Compaction Using Hierarchical Markov Models', Technical Report CENG 97-07, Univ. of Southern California.