

# Vector Compaction Using Dynamic Markov Models\*

Radu MARCULESCU<sup>†</sup>, Diana MARCULESCU<sup>†</sup>, and Massoud PEDRAM<sup>†</sup>, *Members*

**SUMMARY** This paper presents an effective and robust technique for compacting a large sequence of input vectors into a much smaller input sequence so as to reduce the circuit/gate level simulation time by orders of magnitude and maintain the accuracy of the power estimates. In particular, this paper introduces and characterizes a family of dynamic Markov trees that can model complex spatiotemporal correlations which occur during power estimation both in combinational and sequential circuits. As the results demonstrate, large compaction ratios of 1-2 orders of magnitude can be obtained without significant loss (less than 5% on average) in the accuracy of power estimates.

**key words:** power estimation, Markov sources, vector compaction, DMC modeling

## 1. Introduction

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

Power estimation is in general a difficult problem; the key task in this process is the accurate and fast estimation of average switching activity. To date, both simulative [1]–[4] and nonsimulative approaches [5]–[10] have been tried, each one having its own advantages and limitations [11]. More specifically, general simulation techniques provide sufficient accuracy, but at high computational cost; it is simply expensive to simulate thousands of vectors. On the other hand, nonsimulative approaches (best represented by probabilistic power estimation techniques) are in general faster, but less accurate than those based on simulation; usually, the input correlations and the reconvergent fan-out in the target circuit make things very complicated and simplifying assumptions (like input independence) become mandatory.

As a conclusion, a number of issues appear to be important for power estimation and low-power synthesis. The *input statistics* which must be properly captured

and the *length of the input sequences* which must be applied are two such issues. Generating a minimal-length sequence of input vectors that satisfies these statistics is not trivial. The reason is the elaborate set of input statistics that must be preserved or reproduced during sequence generation for use by power simulators. One such attempt is [13] 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. A more elaborate and effective technique was presented in [14] where, based on stochastic sequential machines, the authors succeed in compacting large sequences without significant loss in accuracy. However, in the present research, the limitations of that approach are pointed out and overcome by the proposed technique.

The present paper improves the-state-of-the-art by providing an original solution for vector compaction problem which potentially reduces the gap between simulative and nonsimulative approaches. Having an initial sequence (assumed representative for some target circuit), we target *lossy compression* [15], that is the process of transforming an 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.

The foundation of our approach is probabilistic in nature; it relies on *adaptive (dynamic) modeling* of binary input streams as first-order Markov sources of information and is applicable to combinational and, under some circumstances, to sequential circuits. The adaptive modeling technique itself (best known as *Dynamic Markov Chain* or *DMC modeling*) was introduced very recently in the literature on data compression as a candidate to solve various data compression problems. However, the original model introduced in [17] is not completely satisfactory for our purpose. In this paper, we thus extend the initial formulation to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive input patterns.

As demonstrated and supported by practical evidence, this new framework is extremely effective in power estimation. The basic idea is illustrated in Fig. 1. To evaluate the total power consumption of a target circuit for a given input sequence  $L_0$  (Fig. 1 (a)), we derive

Manuscript received February 25, 1997.

Manuscript revised May 12, 1997.

<sup>†</sup>The authors are with the Department of Electrical Engineering - Systems, University of Southern California, Los Angeles, CA 90089, U.S.A.

\*This research was supported by DARPA under contract F33615-95-C1627, SRC under contract 94-DJ-559, and a grant from Intel Corp.

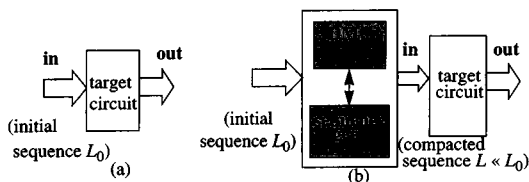


Fig. 1 Data compaction for power estimation.

first the Markov model of the input sequence through a one-pass traversal technique 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. 1 (b)).

We point out here that the present approach can be used without any difficulty to generate benchmark data for power estimation, that is, input sequences with different lengths that satisfy a set of user-prescribed characteristics in terms of word-level transition or conditional probabilities.

To conclude, both simulation-based approaches and probabilistic techniques for power estimation may benefit from this research. The issues brought into attention in this paper are new and represent an important step toward reducing the gap between the simulative and probabilistic techniques commonly used in power estimation. Finally, the concept of DMC modeling itself may find useful applications in other CAD fields.

The paper is organized as follows: Sect. 2 reviews the basic concepts of DMC modeling technique. Section 3 formalizes the power-oriented vector compaction problem and discusses parameters which makes this approach effective in practice. Section 4 presents a DMC-based procedure for vector compaction. In Sects. 5 and 6, we give some practical considerations and experimental results, respectively. Finally, we conclude by summarizing our main contribution.

**2. Background on Dynamic Markov Models**

Without loss of generality, in what follows 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  $k$ -bits. A particular sequence  $S_1$  in  $S$  consists of vectors  $v_1, v_2, \dots, v_n$  (which may be distinct or not), each having a positive occurrence probability. Indices  $1, 2, \dots, n$  represent the discrete time steps when a particular vector is applied to a target circuit. Imposing a total ordering among bits, such a sequence may be conveniently viewed as a binary tree (called  $DMT_0$  from *Dynamic Markov Tree of order zero*) where nodes at level  $j$  correspond to bit  $j$  ( $1 \leq j \leq k$ ) 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

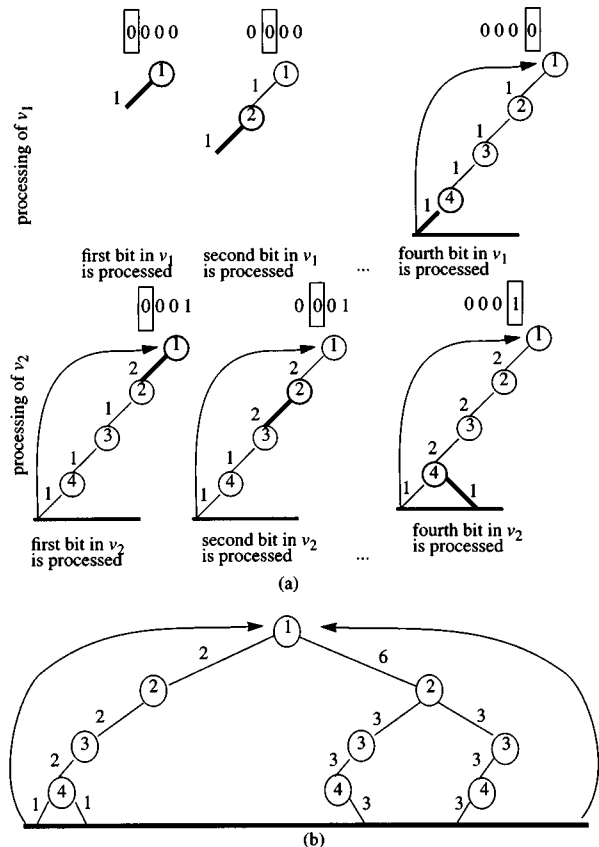


Fig. 2 The tree  $DMT_0$  for the sequence in Example 1.

the root to that particular node, occurred in the original sequence. For clarity, let us consider the following example.

**Example 1:** 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  $DMT_0$  is shown step-by-step in Fig. 2 (a).

Obviously, the whole Markov tree that models this sequence must have four levels because the original sequence is a 4-bit sequence. Without loss of 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  $DMT_0$  as shown in Fig. 2 (a)), the next bit is considered as being bit number two and so on. Every time a vector is completely scanned (this 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 such that, for this particular example, they finally become as indicated in Fig. 2 (b).

The Markov tree in Fig. 2 (b) contains in a compact form all the spatial information about the original sequence  $v_1, v_2, \dots, v_8$ . We point out that this sparse structure is possible only by using the *dynamic (adap-*

tive) fashion of growing the tree  $DMT_0$  just illustrated. Another approach would have been to consider a static binary tree capable to model any 4-bit sequence and just 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 15 instead of 9 nodes in the structure for the same amount of information; this reason alone is sufficient for considering from now on only dynamically grown models.

**Definition 1:** We define the *information source*, to be the pair  $\langle S, p \rangle$ , where  $p$  is a function from  $S$  into  $[0,1]$  satisfying the condition:

$$p(v) = \sum_{x \in S} p(vx) \quad (1)$$

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. 2, the condition (1) is satisfied at every node in this representation<sup>†</sup>. 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 the string '1111' is zero, '1111' being a 'forbidden' vector for this particular sequence.

### 3. Power-Oriented Data Compaction

#### 3.1 Problem Formulation

Input pattern dependencies (that is, spatial and temporal correlations) have a dramatic impact on power dissipation estimates. Spatial correlations refer to the relationship among adjacent bits belonging to the same vector, whereas temporal correlations are the dependencies between successive input patterns. If one ignores the input statistics (which give the actual correlations among the primary inputs), power estimation results can be seriously impaired [7], [11], [16]. In these references, the subject of spatiotemporal correlations was studied from a circuit perspective, that is, the authors consider the impact of the reconvergent fanout on the spatial dependence of internal lines in the circuit. The present paper focuses on the *input problem*, in the sense that it tries to find, *independently* of the target circuit, a good approximation of the input sequence. When applied to the target circuit, this new sequence, must produce the same total power consumption as the initial, but much longer, sequence. Our compaction procedure is completely independent of the target circuit and uses only information about the input sequence. As a consequence, in the present paper, we consider only

spatiotemporal correlations at the primary inputs.

For power estimation purposes, it is therefore critical to distinguish between input sequences which exhibit the same signal probabilities on different bit lines, yet showing very different spatial and temporal correlations, mostly if we are interested in node-by-node power estimation.

Having these issues in mind, the vector compaction problem can be formulated as follows: for a  $k$ -bit sequence of length  $n$  (consisting of vectors  $v_1, v_2, \dots, v_n$ ), find another sequence of length  $m < n$  (consisting of the subset  $u_1, u_2, \dots, u_m$  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 collections of bits) in the original and in the compacted sequence, respectively, the following holds:

$$|p(v^- = X \wedge v^+ = Y) - p(u^- = X \wedge u^+ = Y)| < \varepsilon \quad (2)$$

In relation (2),  $v^-, v^+(u^-, u^+)$  denote the current and the next vector, respectively, in the original (compacted) sequence and  $X, Y$  are any two patterns that appear in the initial sequence. This condition simply requires that the joint transition probability for any group of bits is preserved within a given level of error, for any two consecutive time steps.

#### 3.2 A DMC-Based Approach

An attempt to solve the vector compaction problem for power estimation was recently presented in [14]. In that paper, the authors use elements from probabilistic automata theory to synthesize stochastic machines which can be used in a stand-alone mode for sequence compaction.

From a practical point of view, however, this approach has two inherent limitations:

- The values in the initial transition matrix themselves are important in the decomposition process: some distributions of transition probabilities tend to favor a small number of degenerate matrices, as opposed to others which result in much longer decompositions.
- The compaction technique on stochastic machines is a multiple-step compaction technique. An initial pass through the sequence is performed to extract the statistics of interest; after that, the stochastic machine is synthesized and then the new sequence is generated. This is especially disadvantageous for large sequences when the on-line computer memory and time requirements become prohibitive.

The disadvantages mentioned above can be eliminated by using DMC modeling. To this end, in what follows we introduce an original framework for power-oriented data compaction.

<sup>†</sup>This is actually similar to Kirchoff's law for currents.

From Sect. 3.1, it follows that not only a particular vector  $v_i$  in a given sequence is important, but also its relative position in that sequence matters. More precisely, different permutations of vectors belonging to the same initial set  $(v_1, v_2, \dots, v_n)$ , define completely different input sequences. Coming back to the model presented in Sect. 2, we observe that  $DMT_0$  alone cannot capture this property; we say that  $DMT_0$  has *no memory* and therefore the relative order of vectors in the initial sequence is irrelevant in the construction of  $DMT_0$ . For instance, in Fig. 2 (b), the value of  $3/8$  is the probability to find the particular string (state) '1001' in the original sequence, but this gives no indication about the sequencing of this vector relative to another one, let say '0001.'

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*).

**Example 2:** For the same sequence in Example 1, suppose we want to construct its corresponding tree  $DMT_1$ . We begin as in  $DMT_0$  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  $v_2 = 0001$  follows immediately after  $v_1 = 0000$ ; consequently when we reach the node that corresponds to  $v_1$  (the leftmost path in Fig. 3(a)), instead of going back to the root (and therefore 'forgetting' the context), we start to build a new path (rooted at the current leaf of  $DMT_0$ ) as indicated in Fig. 3(a). The newly constructed path 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. 3(b);  $v_2$  alone leads us to the second leftmost edge of  $DMT_0$  from where, to construct  $DMT_1$ , we have to add the path '1001' which corresponds to  $v_3$ . In this way, we indicate the sequencing between  $v_2$  and  $v_3$  that is,  $v_2 \rightarrow v_3$ .

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 subtree. 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.

Similarly, continuing this process for all leaves in  $DMT_0$ , we end up by building the whole tree  $DMT_1$  as shown in Fig. 4.

In Fig. 4, the upper subtree (levels 1 to 4) represents  $DMT_0$ , 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

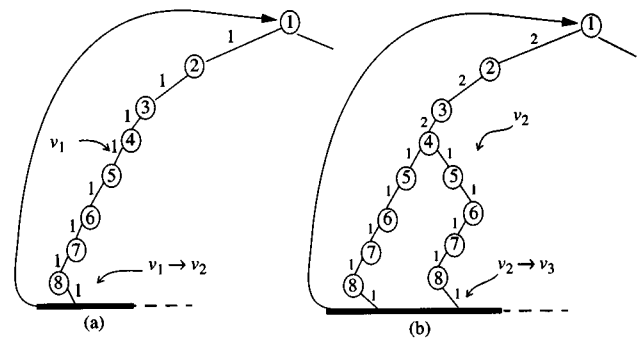


Fig. 3 Construction of the tree  $DMT_1$ .

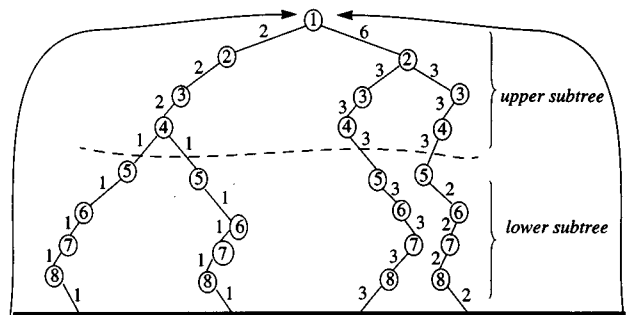


Fig. 4 The tree  $DMT_1$  for the sequence in Example 1.

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 practice the counts of these two subtrees may differ by one (as it can be seen on the rightmost path at the border between the upper and lower subtree in Fig. 4), due to the finite length of the sequences. A practical solution to this issue is to consider the input sequence as being cyclic.

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.

**Proposition 1 [19]:** We write the probability of a vector string  $v = v_1 v_2 \dots v_n$  as follows:

$$p(v) = p(v_1) \cdot p(v_2|v_1) \cdot \dots \cdot p(v_n|v_1 v_2 \dots v_{n-1}) \quad (3)$$

where the conditional probabilities are uniquely defined by:  $p(x|v) = p(vx)/p(v)$ . This property, used in connection with the counts on the edges, allows a quick calculation of the transitions probabilities that characterize any particular sequence. For example, if we want to calculate the transition probability '1001'  $\rightarrow$  '1100' we have from Proposition 1 that  $p(v) = p(v_1 v_2) = p(v_1) \cdot p(v_2|v_1) = 3/8$ , which is exactly the count on the path '10011100' in the tree  $DMT_1$  divided by the sequence length.

**Theorem 1:** Any sequence in  $S$  can be modeled as a first-order Markov source using the structure  $DMT_1$  and parameters  $p$ . We call this process *Dynamic Markov Chain (DMC) modeling*.

**Sketch of proof:** If  $v = v_1v_2$  is a string in the structure  $DMT_1$  such that  $v_1$  is in the upper tree and  $v_2$  is in the lower tree, then  $p(v_2|v_1) = p(v)/p(v_1)$ . Thus, the parameters stored on the edges of  $DMT_1$  structure provide the conditional probabilities that characterize the lag-one Markov chain for the sequence in  $S$ .  $\square$

**Theorem 2:** The structure  $DMT_1$  and parameters  $p$  are equivalent to a stochastic sequential machine.

**Sketch of proof:**  $DMT_1$  defines a Markov source (based on Theorem 1). Any Markov source is characterized by a stochastic matrix  $A$ . According to the decomposition Theorem 1 given in [14], this matrix is uniquely associated to a stochastic machine (a finite-state machine with randomly generated inputs). Thus,  $DMT_1$  is equivalent to a SSM.  $\square$

Generally speaking, the theory of stochastic sequential machines is far more developed than the theory of DMC modeling. However, the DMC modeling technique based on  $DMT_1$  seems to be more effective as it offers a much more compact structure and generally outperforms the compaction techniques based on stochastic machines. Specifically:

- Sequence compaction based on  $DMT_1$  avoids the time consuming decomposition process necessary in stochastic machines' synthesis.
- Using  $DMT_1$  one can avoid the need for partitioning the input vectors into groups of bits. Therefore, one can

expect to improve the accuracy, especially in those cases when the input patterns are highly correlated.

- $DMT_1$  is constructed dynamically (new nodes are added only 'on demand') therefore it offers a much more compact data structure than matrix  $A$  does.

The structure  $DMT_1$  just introduced is general enough to capture completely spatial correlations and first-order temporal correlations. 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 two successive input patterns.

#### 4. A DMC-Based Vector Compaction Procedure

A practical procedure to construct  $DMT_1$  and generate the compacted sequence is given in Fig. 5(a). During a one-pass traversal of the original sequence (when we extract the bit-level statistics of each individual vector  $v_1, v_2, \dots, v_n$  and also those statistics that correspond to pairs of consecutive vectors  $(v_1v_2), (v_2v_3), \dots, (v_{n-2}v_{n-1}), (v_{n-1}v_n)$ ) we grow simultaneously the tree  $DMT_1$ . We continue to grow  $DMT_1$  as long as the number of nodes in the

```

procedure DMC (input_file, ratio, model_size) {
  initial_state = new_state ();
  symbol = read_input (input_file);
  update_tree (symbol, upper_tree, initial_state);
  crt_state = last_state (symbol, upper_tree);
  while (!EOF (input_file)) {
    symbol = read_input (input_file);
    if (number_of_states < model_size) {
      update_tree (symbol, lower_trees, crt_state);
      update_tree (symbol, upper_tree, initial_state);
      crt_state = last_state (symbol, upper_tree);
    }
    else {
      generate_seq (upper_tree, lower_trees, ratio);
      flush_model (upper_tree, lower_trees);
      initial_state = new_state ();
      symbol = read_input (input_file);
      update_tree (symbol, upper_tree, initial_state);
      crt_state = last_state (symbol, upper_tree);
    }
  }
  generate_seq (upper_tree, lower_trees, ratio);
}

```

(a)

```

procedure generate_seq (upper_tree, lower_trees, ratio)
{
  crt_symbol = generate_random ();
  lower_tree_node = last_node (upper_tree, crt_symbol);
  upper_tree_node = root (upper_tree);
  do {
    for each bit in the current vector {
      generate '0' or '1' to maximize the decrease in
      absolute error;
      if ('0' is generated) {
        lower_tree_node = left (lower_tree_node);
        upper_tree_node = left (upper_tree_node);
      }
      else {
        lower_tree_node = right (lower_tree_node);
        upper_tree_node = right (upper_tree_node);
      }
    }
    lower_tree_node = upper_tree_node;
    upper_tree_node = root (upper_tree);
  }
  while there are still vectors to be generated;
}

```

(b)

Fig. 5 The vector compaction procedure.



In the second scenario, illustrated in Fig. 7, the *modelSize* parameter is set by the user as being 30 therefore the tree in Scenario 1 cannot be grown as such because the limit of 30 nodes is reached before the whole sequence is scanned. As a consequence, once we reach this limit (this actually happens immediately after processing the subsequence  $v_1, v_2, \dots, v_9$ ), we stop growing the tree and call *generate\_seq* procedure with parameter *ratio* = 2 (Fig. 7 (a)). This will produce a subsequence of 4 vectors which best approximate the first 'segment' ( $v_1, v_2, \dots, v_9$ ) of the original sequence. After that we flush the model (keeping only the very last processed vector  $v_9$ ) and start a new Markov tree as shown in Fig. 7 (b). When the whole sequence is exhausted, based on this new Markov tree, we generate a new subset of 4 vectors which best approximate the second 'segment' ( $v_9, v_{10}, v_{11}, \dots, v_{17}$ ) of the original sequence.

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. This is an essential capability needed to avoid 'hang-up' ('forbidden') states of the circuit during simulation process for power estimation.

In general, by alternating the generation and flushing phases in the DMC procedure, the complexity of the model can be effectively handled. The issue of accuracy in the context of these repeated flushes is discussed in the subsequent section.

## 5. Practical Considerations

### 5.1 Complexity Related Issues

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  $DMT_1$ , while the input sequence is scanned incrementally, both the set of states and the transition probabilities change dynamically making this technique highly adaptive.

A natural question still remains: when should the growing process be halted? If it is not halted, there is no bound on the amount of memory needed. On the other side, if it is completely halted we lose the ability to adapt if some characteristics of the source message change. A practical solution is to set a limit on the number of states in the DMC [17] as we actually did in [12]. When this limit is reached, the Markov model is flushed and a new model is started. Although this solution may appear as too drastic, in practice it performs very well. The intuition behind this property is the capability of DMC model to adapt very fast to changes that occur while the input is scanned. A less extreme solution to limit model growing is also possible; we can keep a backup buffer that retains the last  $p$  vectors

emitted by the source and whenever the model should be discarded, we may reuse this information to avoid starting the new model from the scratch.

### 5.2 Accuracy Related Issues

Let us consider first the case when the number of nodes allowed in the model is sufficiently large to process the whole sequence. In this case, we may use two compaction strategies. The first is to monitor the current values of the transition probabilities and compare them with the transition probabilities of the original sequence. When the difference between the two sets of probabilities becomes sufficiently small, the generation process is halted. Therefore, we are able in this way to satisfy any user specified error level for the transition probabilities. The second strategy is to set the compaction ratio to a given value, perform compaction, and then compute the error obtained by compaction a posteriori.

If the number of cells allowed in the model (*modelSize* parameter in DMC procedure) is too small to allow the processing of the whole sequence, then we must resort to flushing. We should note that in this case the same compaction ratio has to be used for all subsequences and thus its value has to be set upfront. To see how the flushing technique affects the accuracy, 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  $n_1 + n_2 + \dots + n_f = n$ .

**Theorem 3 [12]:** If the  $i$ -th subsequence is approximated with an error less than  $\varepsilon_i$ , then the accuracy for the whole sequence is:

$$\varepsilon = (1/n) \cdot \sum_{i=1}^f n_i \cdot \varepsilon_i \leq \max(\varepsilon_i) \quad (4)$$

where  $r$  is the compaction ratio.

Therefore, as long as the models for partial DMCs capture the transition probabilities for the initial subsequences up to some  $\varepsilon_i$ , then the transition probabilities for the entire sequence are preserved up to some  $\varepsilon$ .

However, the non-homogeneous sequences that may arise in practice (e.g. sequences which exhibit widely different transition behaviors over time) can have very different transition probabilities for each of their subsequences. In such cases, if flushing is done properly so as to distinguish between subsequences with different transition behavior, then the overall accuracy can be significantly improved. This is an interesting problem which requires further investigation.

## 6. Experimental Results

The overall strategy is depicted in Fig. 8. We assume that the input data is given in the form of a sequence

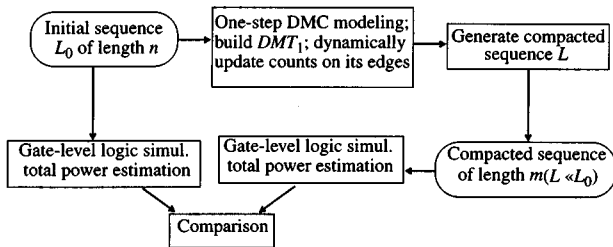


Fig. 8 Experimental setup.

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  $DMT_1$ ; during this process, the frequency counts on edges of  $DMT_1$  are dynamically updated.

The next step in Fig. 8 does the actual generation of the output sequence (of length  $m$ ). If the initial sequence has the length  $n$  and the new generated sequence has the length  $m < n$ , then we say that a *compaction ratio* of  $r = n/m$  was achieved.

Finally, a validation step is included in the strategy; for short sequences we used the commercial tool PowerMill [2] whilst for long sequences we resorted to an in-house gate-level logic simulator developed under SIS.

In Tables 1, 2, we provide the real-delay results for two types of initial sequences. Sequences of type 1 are large input streams having the same initial length  $n = 100,000$  and being then prime candidates for compaction; type 1 refers to biased sequences obtained by doing bit-level logical operations on ordinary pseudo-random sequences. The sequences of type 2 (having the length 4,000) are highly biased sequences obtained from real industry applications.

As shown in Table 1, sequences of type 1 were compacted with two different compaction ratios (namely  $r = 50$  and  $100$ ); we give in this table the total power dissipation measured for the initial sequence (column 3) and for the compacted sequence (columns 4, 5). In the last column, we give the time in seconds (on a Sparc 20 workstation with 64 Mbytes of memory) necessary to read and compress data with DMC modeling.

Since the compaction with DMC modeling is linear in the number of nodes in the structure  $DMT_1$ , the values reported in the last column are far less than the actual time needed to simulate the whole sequence. During these experiments, the number of states allowed in the Markov model was 20,000 (about 560 kbytes).

As we can see, the quality of results is very good even when the length of the initial sequence is reduced by 2 orders of magnitude. Thus, for C432 in Table 1, instead of simulating 100,000 vectors with an exact power of  $1816.32 \mu\text{W}$ , one can use only 2,000 vectors with an estimate of  $1838.89 \mu\text{W}$  or just 1,000 vectors with a power consumption estimated as  $1779.60 \mu\text{W}$ . This re-

Table 1 Total power ( $\mu\text{W}$  @ 20 MHz) for sequences of type 1.

Circuit	No. of Inputs	Power for initial seq.	Power for $r = 50$	Power for $r = 100$	Time for DMC (sec)
C432	36	1816.32	1838.89	1779.60	42
C499	41	3697.84	3546.65	3622.26	48
C880	60	3314.07	3229.85	3329.31	75
C1355	41	3205.27	3044.20	3109.18	48
C3540	50	10876.22	9910.08	10687.32	61
C6288	32	110038.69	114199.50	109077.42	37
s344	9	751.58	748.54	719.53	10
s386	7	818.11	844.58	848.80	8
s510	19	2269.97	2354.08	2337.59	17
s820	18	3996.17	4028.22	4049.03	17
s838	34	1052.05	1061.73	1091.14	41
s953	16	1479.25	1492.78	1468.48	16
s1196	14	3687.47	3702.32	3580.63	16
s5378	35	12781.99	12507.97	12609.96	41
s9234	36	9192.75	9157.31	9209.75	43
		% error	2.55	2.57	

Table 2 Total current (mA) for sequences of type 2.

Circuit	No. of Inputs	Initial sequence		Compacted sequence		
		Current (mA)	Time to simulate (sec)	Current (mA) $r = 5$	Current (mA) $r = 10$	Time to simulate (sec) $r = 10$
C432	36	0.4135	1186	0.4352	0.4404	120
C499	41	0.8188	2675	0.8337	0.8290	235
C880	60	0.7907	2289	0.8324	0.8023	274
C1355	41	1.1375	2993	1.1549	1.1461	284
C1908	33	1.2976	4034	1.2821	1.2833	367
C3540	50	3.4490	9467	4.0500	3.8580	1082
C6288	32	14.5749	88032	14.8020	15.9315	5005
		% error		4.85	4.80	

duction in the sequence length has a significant impact on speeding-up the simulative approaches where the run time is proportional to the length of the sequence which must be simulated.

The sequences of type 2 were compacted for two compaction ratios ( $r = 5$  and  $r = 10$ ) using PowerMill; to assess the potential of efficiency of the approach, for both original and compacted sequences, we report also the actual run time required by PowerMill to provide power estimates. The number of nodes allowed for the Markov model construction, was 5,000 (about 140 kbytes); the CPU time for DMC modeling was below 3 seconds in all cases.

As it can be seen in Table 2, the average relative error is below 5% while the speed-up in power estimation is about one order of magnitude on average. For example, using the original sequence of 4,000 vectors, PowerMill took for C432 about 1,186 seconds to estimate a total current of 0.4135 mA. On the other side, using the sequence generated with DMC of only 400 vectors ( $r = 10$ ), PowerMill estimated a total current of 0.4066 mA in only 120 seconds.

We note also, that the results presented both Tables 1 and 2, are significantly better than those reported in [14] in terms of running time and memory require-

**Table 3** Results for simple random sampling.

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

**Table 4** Results for DMC approach.

Circ.	No. of vect.	Error violations		
		> 5%	> 6%	>10%
C432	2000	6.7	1.9	0.0
C499	800	0.3	0.0	0.0
C880	2000	1.4	0.1	0.0
C1355	800	0.2	0.0	0.0
C1908	800	1.9	1.2	0.0
C3540	1000	0.9	0.0	0.0
C6288	2000	0.0	0.0	0.0

ments.

Finally, we compare our results with simple random sampling 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<sup>†</sup>. We report in Table 3 the maximum and average number of vector pairs needed for total power values to converge [11]. We also indicate the percentage of error violations for total power values, using as thresholds 5%, 6% and 10%. Using different seeds for the random number generator (and therefore choosing different initial states in the sequence generation phase), we run a set of 1,000 experiments for the DMC technique. In Table 4, we give the DMC results for the same thresholds as those used in simple random sampling.

Once again, the results obtained with DMC modeling technique score very well and prove the robustness of the present approach. As we can see, using fewer vectors, the accuracy of DMC is higher than the one of simple random sampling in most of the cases.

## 7. Conclusion

In this paper, we addressed the vector compaction problem from a probabilistic point of view. Based on dynamic Markov Chain modeling, we proposed an original approach to compact an original 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.

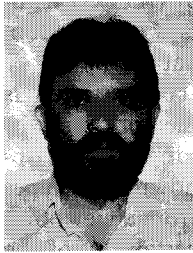
<sup>†</sup>This means that the probability of having a relative error larger than 5% is only 1%.

The mathematical foundation of this approach relies in Markov models; within this framework a family of dynamic Markov trees is introduced and characterized as an effective and flexible way to model complex spatiotemporal correlations which occur during power estimation. The results obtained both on combinational and sequential benchmarks show that large compaction ratios of 1–2 orders of magnitude can be obtained without much loss in accuracy in total power estimates.

## References

- [1] S.M. Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE J. Solid-State Circuits*, vol.21, no.5, pp.889–891, Oct. 1986.
- [2] C.X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, "The design and implementation of powermill," *Proc. Intl. Workshop on Low Power Design*, pp.105–110, April 1995.
- [3] B.J. George, D. Gossain, S.C. Tyler, M.G. Wloka, and G.K. Yeap, "Power analysis and characterization for semicustom design," *Proc. Intl. Workshop on Low Power Design*, pp.215–218, April 1994.
- [4] F.N. Najm, "A Monte Carlo approach for power estimation," *IEEE Trans. VLSI Systems*, vol.1, no.1, pp.63–71, March 1993.
- [5] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," *Proc. ACM/IEEE Design Automation Conference*, pp.253–259, June 1992.
- [6] F.N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.*, vol.12, no.2, pp.310–323, Feb. 1993.
- [7] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient power estimation for highly correlated input streams," *Proc. ACM/IEEE Design Automation Conference*, pp.628–634, June 1995.
- [8] A. Chandrakasan, et al., "HYPER-LP: A system for power minimization using architectural transformation," *Proc. IEEE/ACM Intl. Conference on Computer Aided Design*, pp.300–303, Nov. 1992.
- [9] P. Landman and J. Rabaey, "Power estimation for high level synthesis," *Proc. European Design Automation Conference*, pp.361–366, Feb. 1993.
- [10] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures for power analysis," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.*, vol.15, no.6, June 1996.
- [11] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Trans. Design Automation of Electronic Systems*, vol.1, no.1, pp.1–54, Jan. 1996.
- [12] R. Marculescu, D. Marculescu, and M. Pedram, "Vector compaction using dynamic Markov models," *Technical Report CENG 96-14*, Univ. of Southern California, Feb. 1996.
- [13] J. Monteiro and S. Devadas, "Techniques for power estimation of sequential logic circuits under user-specified input sequences and programs," *Proc. Intl. Workshop on Low Power Design*, pp.33–38, April 1994.
- [14] D. Marculescu, R. Marculescu, and M. Pedram, "Stochastic sequential machine synthesis targeting constrained sequence generation," *Proc. ACM/IEEE Design Automation Conference*, pp.696–701, June 1996.
- [15] T. Bell, J. Cleary, and I. Witten, "Text Compression," Prentice Hall, 1990.

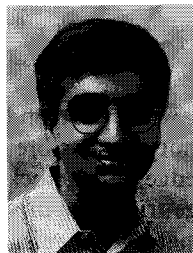
- [16] R. Marculescu, D. Marculescu, and M. Pedram, "Switching activity analysis considering spatiotemporal correlations," Proc. IEEE/ACM Intl. Conference on Computer Aided Design, pp.294-299, Nov. 1994.
- [17] G.V. Cormack and R.N. Horspool, "Data compression using dynamic Markov modeling," Computer Journal, vol.30, no.6, pp.541-550, 1987.
- [18] A. Davis, "Markov chains as random input automata," American Mathematical Monthly, vol.68, pp.264-267, 1961.
- [19] A. Papoulis, "Probability, Random Variables, and Stochastic Processes," McGraw-Hill Co., 1984.
- [20] J.W. Green and K.J. Supowit, "Simulated annealing without rejected moves," Digest. of Intl. Conference on Computer Design, pp.658-663, Oct. 1984.



**Radu Marculescu** received the M.S. degree in Electrical Engineering from Polytechnic Institute of Iasi, Romania, in 1985. Currently, he is pursuing the Ph.D. degree in Computer Engineering at Univ. of Southern California, Department of EE-Systems. Prior to USC he was with Computer Science Dept. at Polytechnic Institute of Bucharest. His research interests include CAD of VLSI with emphasis on low-power design methodologies.



**Diana Marculescu** received the M.S. degree in Computer Science from Polytechnic Institute of Bucharest, Romania, in 1991. After graduation, she was with the Computer Science Department in Bucharest until 1993. Currently, she is a Research Assistant at the Univ. of Southern California in the Department of EE-Systems, working towards the Ph.D. degree in Computer Engineering. Her research interest include hardware and software issues in low-power systems.



**Massoud Pedram** received the B.S. degree in Electrical Engineering from California Institute of Technology in 1986 and the M.S. and Ph.D. degrees from University of California, Berkeley, in 1989 and 1991, respectively. Currently, he is an Associate Professor of electrical engineering at the Univ. of Southern California, Los Angeles. His research interest span many aspects of design and synthesis of VLSI circuits, with particular emphasis

on layout-driven synthesis and design for low-power.