Adaptive Models for Input Data Compaction for Power Simulators*

Radu Marculescu, Diana Marculescu, Massoud Pedram

Department of Electrical Engineering - Systems
University of Southern California, Los Angeles, CA 90089
e-mail: {radu,draiciu,massoud}@danube.usc.edu

Abstract - 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.

I. 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 in 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 both to combinational and 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 [16] 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.1a), we derive 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.1b).

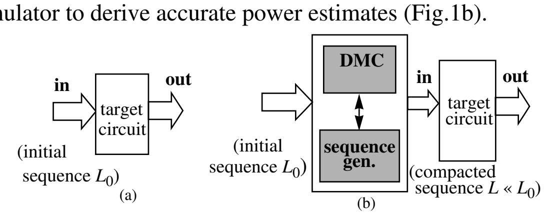

Fig.1

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

II. 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,..., v_n (which may be distinct or not), each having a positive occurence probability. Indices 1, 2,..., n represent the discrete time steps when a particular vector is applied to a target circuit. Imposing a total

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

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 ($1 \le j \le k$) 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 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.2a. 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.2a), 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 those indicated in Fig.2b.

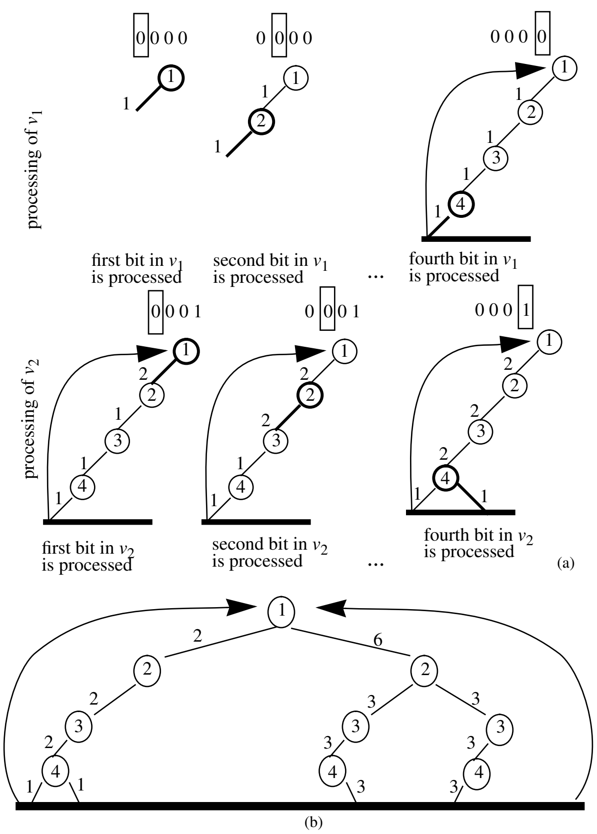

Fig.2

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

$$P(v) = \sum_{x \in S} P(vx) \qquad (1)$$

for all v in S, where vx represents the event corresponding to the joint occurence 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, condition (1) is satisfied at every node in this representation[2]. In addition, based on the counts of the terminal edges, we may easily compute the probability of occurence for a particular vector in the sequence. For instance, the probability of occurence 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.

III. Power-Oriented Data Compaction

A. Problem Formulation

Input pattern dependence has a dramatic impact on power dissipation estimates. If one ignores the input statistics (which give the actual correlations among the primary inputs), power estimation results can be seriously impaired.

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 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 $v_1, v_2, \ldots, v_n$), find another sequence of length $m < n$ (consisting of the subset $u_1, u_2, \ldots, u_m$ of the initial sequence), such that the a*verage 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:

$$\left| P(v^- = X \wedge v^+ = Y) - P(u^- = X \wedge u^+ = Y) \right| < \varepsilon \qquad (2)$$

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.

B. A DMC-based Approach

An attempt to solve the vector compaction problem for power

[2]This is actually similar to Kirchoff's law for currents.

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.

From Section III.A, 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,..., v_n), define completely different input sequences. Coming back to the model presented in Section II, 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. In Fig.2b for instance, the value of 3/8 is the probability that we find the particular string (state) '1001' in the original sequence, but this gives us no indication at all about the sequencing of this vector relative to another one, 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*).

<u>Example 2:</u> 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.3a), 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.3a. Basically, we added a new path that corresponds to '0001'. 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.3b.

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* 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 first-order Markov sources of information.

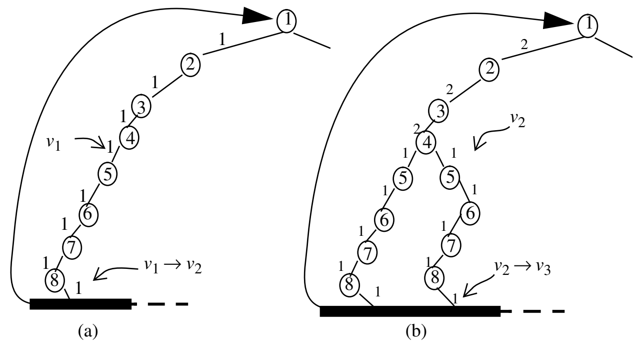

Fig.3

Similarly, continuing this process for all leaves in DMT_0 in Fig.2b, we end up by building the whole tree DMT_1 as shown in Fig.4.

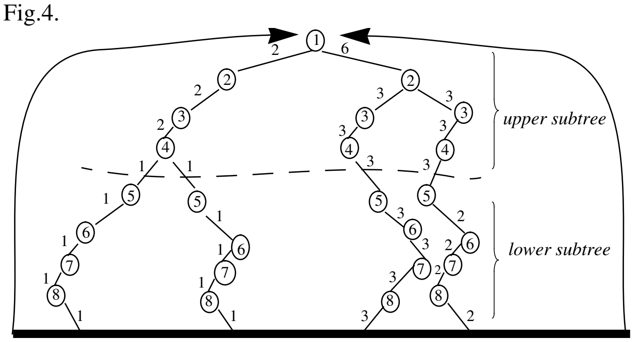

Fig.4

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 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, 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 $v = v_1 v_2 ... v_n$ as follows:

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

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 $P(v) = P(v_1 v_2) = P(v_1) \cdot P(v_2 | v_1) = 3/8$ 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.

Theorem 2. The structure DMT_1 and parameters P are equivalent to a stochastic sequential machine. (Proofs can be found in [12]).

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.

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.

IV. A DMC-based Vector Compaction Procedure

A practical procedure to construct DMT_1 and generate the compacted sequence is described subsequently. 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 pairs of consecutive vectors (v_1v_2), $(v_2v_3),...,(v_{n\text{-}2}v_{n\text{-}1}),(v_{n\text{-}1}v_n))$ we grow simultaneously the tree DMT_1. We continue to grow DMT_1 as long as the number of nodes in the Markov model is smaller than a user-specified threshold. After reaching the threshold we generate the new sequence up to that point and discard (flush) the model; a detailed example involving flushes is worked out in [12]. A new Markov model is started again and the process is continued up to the end of the original sequence. 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.

Each generation phase is driven by the user-specified compaction parameter *ratio* that is, in order to generate a total of $m = n/ratio$ vectors, we have to keep the same compaction ratio for every dynamically grown Markov model. For generation, 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 decreased and the symbols are generated according to their probability distribution. We use this strategy only to generate the first vector. After that, to ensure a minimal level of error, we use an *error controlling mechanism*. The pseudocode for the generation phase and a detailed example is given in [12].

In all our experiments we used the DMC modeling technique based on the structure DMT_1. 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 circuit during simulation process for power estimation.

IV. Practical Considerations

A. 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.

Input sequences having a large number of bits k 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_1 will be. The DMC modeling technique exploits this observation by starting with an initially empty model and dynamically growing (’on-demand’) 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.
- 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.

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.

B. Accuracy Related Issues

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 $n_1 + n_2 + ... + n_f = n$. Let v_i (u_i) be a vector from the initial (compacted) i-th subsequence (obtained due to successive flushes) and v (u) a vector from the initial (compacted) sequence.

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

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

where r is the compaction ratio.

Therefore, as long as the models for partial DMCs accurately capture the transition probabilities for the initial subsequences, the transition probabilities for the entire sequence are preserved up to some ε. However, the non-homogeneous sequences that may arise in practice (e.g. sequences with bi-modal distribution) can have very different transition probabilities for each subsequence. 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.

V. Experimental Results

The overall strategy is depicted in Fig.5. 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_1; during this process, the frequency counts on edges of DMT_1 are dynamically updated.

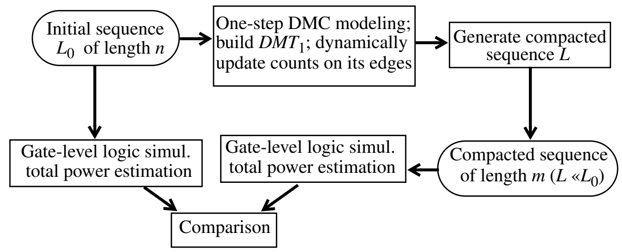

Fig.5

The next step in Fig.5 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 only 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 pseudorandom 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.

Table 1: Total Power (uW@20MHz) 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
s838	34	1052.05	1061.73	1091.14	41
s1196	14	3687.47	3702.32	3580.63	16
s9234	36	9192.75	9157.31	9209.75	43
		% error	2.80	2.93	

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.

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 uW, one can use only 2000 vectors with an estimate of 1838.89 uW or just 1000 vectors with a power consumption estimated as 1779.60 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.

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

Table 2: Total Current (mA) for sequences of type 2

		Initial sequence		Compacted sequence		
Circuit	No.of Inputs	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	

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 4000 vectors, PowerMill took for C432 about 1186 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 requirements.

Finally, we compare our results with simple random sampling of vector pairs from the original sequences [21]. 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 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.

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

Table 3: Simple Random Sampling

	Number of vector pairs		Error violations		
Circ.	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: DMC Approach

		Error violations		
Circ.	No. of vect.	> 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

V. 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.

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.

Acknowledgement

The authors would like to thank to C.S. Ding of USC for helping with results in Table 3.

References

[1] S.M.Kang, 'Accurate Simulation of Power Dissipation in VLSI Circuits', in *IEEE Journal of Solid State Circuits,* 21 (5), pp. 889-891, Oct.1986.

[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] B.J.George, D.Gossain, S.C.Tyler, M.G.Wloka, and G.K.Yeap, 'Power Analysis and Characterization for Semi-Custom Design', in *Proc. Intl. Workshop on Low Power Design,* pp.215-218, April 1994.

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

[5] A. Ghosh, S. Devadas, K. Keutzer, and J. White, 'Estimation of Average Switching Activity in Combinational and Sequential Circuits', in *Proc. ACM/IEEE Design Automation Conference,* pp. 253-259, June 1992.

[6] 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.

[7] 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.

[8] A. Chandrakasan, et. al, 'HYPER-LP: A System for Power Minimization Using Architectural Transformation', in *Proc. IEEE/ACM Intl. Conference on Computer Aided Design,* pp. 300-303, Nov.1992.

[9] P. Landman, J. Rabaey, 'Power Estimation for High Level Synthesis', in *Proc. European Design Automation Conferenc*e, pp. 361-366, Feb.1993.

[10] D. Marculescu, R. Marculescu, and M. Pedram, 'Information Theoretic Measures for Energy Consumption at Register Transfer Level', in *Proc. Intl. Workshop on Low Power Design,* pp. 81-86, April 1995.

[11] 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.

[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', in *Proc. Intl. Workshop on Low Power Design,* pp. 33-38, April 1994.

[14] 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.

[15] J. Storer, 'Data Compression: Methods and Theory', Ch.1, Computer Science Press, 1988.

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

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

[18] A. Davis, 'Markov Chains as Random Input Automata', in *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', in *Digest. of Intl. Conference on Computer Design,* pp. 658-663, Oct. 1984.

[21] I.R. Miller, J.E. Freund and R. Johnson, 'Probability and Statistics for Engineers', Prentice Hall, 1990.