Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation*

Diana Marculescu, Radu Marculescu, Massoud Pedram

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

Abstract - The problem of stochastic sequential machines (SSM) synthesis is addressed and its relationship with the constrained sequence generation problem which arises during power estimation is discussed. In power estimation, one has to generate input vector sequences that satisfy a given statistical behavior (in terms of transition probabilities and correlations among bits) and/or to make these sequences as short as possible so as to improve the efficiency of power simulators. SSMs can be used to solve both problems. Based on Moore-type machines, a general procedure for SSM synthesis is revealed and a new framework for sequence characterization is built to match designer's needs for sequence generation or compaction. As results demonstrate, compaction ratios of 1-2 orders of magnitude can be obtained without much loss in accuracy of total power estimates.

I. Introduction

With the growing need for low-power electronic circuits and systems, power analysis and low-power synthesis have become crucial tasks that must be addressed in order to design complex high-performance digital systems. Power estimation techniques must be fast and accurate in order to be applicable in practice. Not surprisingly, these two requirements interfere with one another and at same point they become contradictory.

General simulation techniques can provide high accuracy, but at high computational cost. It is impractical to simulate large circuits using millions or even thousands of input vectors and therefore, the length of the sequence to be simulated is an important consideration.

Probabilistic power estimation techniques generally have a low computational overhead, but tend to be less accurate [1]. A major concern in probabilistic approaches is the ability to account for internal dependencies due to the reconvergent fan-out in the circuit. This problem, which we will refer to as 'the circuit problem', is by no means trivial. Indeed, a whole set of solutions have been proposed, ranging from approaches which build the global OBDD [2] and therefore capture all internal dependencies, to intricate techniques which partially account for dependencies in an incremental (and therefore more efficient) manner [3][4] or use local OBDDs [5][6]. Recently, we pointed out the importance of correlations not only inside the target circuit, but also at the circuit inputs [7]. We will refer to this as 'the input problem'; it is important not only in power estimation, but also in low-power design.

Generating a minimal-length sequence of input vectors that satisfies a prescribed set of statistics in not trivial. One such attempt is made in [9] where authors use deterministic FSMs to model user-specified input sequences. Since the number of states in the FSM is equal to the number of patterns in the sequence to be modeled, the ability to characterize anything other than short input sequences is limited.

In summary, 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 sequence* which must be applied, are two such issues. From this perspective, the present paper shifts the focus from 'the circuit problem' to 'the input problem' and improves the state-of-the-art by proposing an original solution for *constrained sequence generation*[1].

Over the years, many important problems in sequential circuit synthesis and optimization have been approached using concepts from automata theory. It is quite natural (and useful) to consider automata with stochastic behavior. The idea is that the automaton, when in state s_i and receiving input x, can move into any new state s_j with a positive probability $p(s_i,x)$. A practical motivation for considering probabilistic automata is that even sequential circuits which are intended to behave deterministically, exhibit stochastic behavior because of random malfunctioning of components [10].

The mathematical foundation of our approach relies on the stochastic sequential machines (SSM) theory and without any loss in generality, emphasizes those aspects related to Moore-type machines which are applicable to both combinational and sequential circuits. We reveal a general procedure for SSM synthesis and describe a new framework for sequence characterization to match designer's needs for sequence generation or compaction. We focus our attention to the basic task of synthesizing an SSM, able to generate constrained input sequences. Such a machine can be used not only in the stand-alone mode (as is the case for sequence compaction) but also in the front of the target circuit (for probabilistic power estimation).

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 a first step toward reducing the gap between the simulative and probabilistic techniques commonly used in power estimation.

The paper is organized as follows. Section II introduces some

*This research was supported by ARPA under contract F33615-95-C1627, SRC under contract 94-DJ-559, NSF under contract MIP-9457392 and a grant from Toshiba Corp.

[1] Simply stated, any input sequence that must satisfy a set of spatial and/or temporal correlations is 'constrained'.

basic definitions from SSM theory and gives the main decomposition theorems used in SSM synthesis. Section III discuss the constrained sequence generation problem and gives a practical procedure for sequence compaction. Section IV is devoted to practical considerations and experimental results. Finally, we conclude by summarizing our main contribution.

II. The Synthesis of SSM

In this section, we review the concept of SSM and describe a basic procedure for synthesizing SSMs from their mathematical models.

A. Stochastic machines: basic definitions

Definition 1. (Mealy-type SSM)

A Mealy-type SSM is a quadruple $\mathcal{M} = (S, X, Y, \{A(x, y)\})$ where S, X, and Y are finite sets (the internal states, inputs, and outputs, respectively), and $\{A(x, y)\}$ is a finite set containing $|X| \times |Y|$ square stochastic matrices of order $|S|$ such that $a_{ij}(y|x) \geq 0$ for all i and j, and

$$\sum_{y \in Y} \sum_{j=1}^{|S|} a_{ij}(y|x) = 1 \quad \text{where} \quad A(y|x) = a_{ij}(y|x) \qquad (1)$$

Interpretation: Let π be any $|S|$-dimensional vector. If the machine begins with an initial distribution π over the state set S and is fed with a word x, it outputs the word y and moves on to the next state. The transition is controlled by the *transition matrix* $A(y|x)$ where $a_{ij}(y|x)$ is the *conditional probability* of the machine going to state s_j and producing the symbol y, given it had been in state s_i and fed with symbol x.

Definition 2.

Let $\mathcal{M}$ be an SSM, $u = x_1x_2...x_k$ an input sequence and $v = y_1y_2...y_k$ an output sequence. By definition, $A(v|u) = [a_{ij}(v|u)] = A(y_1|x_1) \cdot A(y_2|x_2) \cdot ... \cdot A(y_k|x_k)$; it follows from the interpretation of the values of $a_{ij}(y|x)$ that $a_{ij}(v|u)$ is the probability of machine going to state s_j and producing the sequence v, having been in state s_i and fed sequentially the sequence u.

Example: Let $M = (S, X, Y, \{A(y|x)\})$ with $X = \{0, 1\}$, $Y = \{a, b\}$, $S = \{s_1, s_2\}$ and

$$A(a|0) = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \quad A(b|0) = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 \end{bmatrix} \quad A(a|1) = \begin{bmatrix} 0 & \frac{1}{2} \\ 0 & 0 \end{bmatrix} \quad A(b|1) = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and let p = $\begin{pmatrix} \frac{1}{4} & \frac{3}{4} \end{pmatrix}$ be an initial distribution for M. First, we can see that M is correctly defined, that is equation (1) is verified for every possible initial state. Inspecting for instance the first row in matrices $A(a|0)$ and $A(b|0)$, we observe that machine M, initially in state s_1 and fed with symbol 0, will remain in state s_1 and produce a symbol a with probability 1/2, will remain in state s_1 and produce a symbol b with probability 1/4, or will go to state s_2 and generate a symbol b with probability 1/4. In addition, from definition 2 we have that

$$A(ab|00) = A(a|0) \cdot A(b|0) = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} \\ \frac{1}{4} & 0 \end{bmatrix} .$$

This means for instance, that the probability of the machine printing the word ab, having been in state s_1 and fed the word 00 on the input, is $\frac{1}{8} + \frac{1}{8} = \frac{1}{4}$ (irrespective of the final state of the machine).

Definition 3. (Moore-type SSM)

A Moore-type SSM is a quintuple $\mathcal{M} = (S, X, Y, \{A(x)\}, \Lambda)$ where S, X, and Y are as in Definition 1, $\{A(x)\}$ is a finite set containing $|X|$ square stochastic matrices of order $|S|$ and Λ a deterministic function from S into Y.

Interpretation: The value $a_{ij}(x)$ ($A(x) = [a_{ij}(x)]$) is the probability of the machine moving from state s_i to s_j when fed with the symbol x. When entering state s_j, the machine outputs the symbol $\Lambda(s_j) \in Y$.

Definition 4.

Let $\mathcal{M}$ be an SSM. Let $A(u) = [a_{ij}(u)] = A(x_1) \cdot A(x_2) \cdot ... \cdot A(x_k)$; it follows from the above interpretation that $a_{ij}(u)$ is the probability of the machine going from state s_i to state s_j when fed the word u. The output word v depends on the sequence of states through the machine passed when scanning the input word u.

As we can see from the above definitions, Mealy and Moore stochastic machines generalize the corresponding definitions of deterministic machines. We should note that Mealy-Moore equivalence is still valid for stochastic machines, that is every Moore-type SSM has a Mealy-type equivalent and vice versa.

B. The synthesis procedure

Without loss of generality, in what follows the machines are assumed to be of Moore-type. The basic procedure can be simplified by means of the following important result.

Theorem 1. [11]: Any $m \times n$ stochastic matrix A can be expressed in the form $A = \sum p_i U_i$ where $p_i > 0$, $\sum p_i = 1$, and U_i are *degenerate* stochastic matrices (i.e., having only 0 or 1 entries), and the number of matrices U_i in the expansion is *at most* $m(n\text{-}1) + 1$. ■

The theorem we provide next is very important from a practical point of view; it gives the basis to efficiently apply Theorem 1 on large matrices which may arise in practice.

Theorem 2. The sequence $\{p_i\}_{i \geq 1}$ is monotonically non-increasing and strictly positive.

Proof: It suffices to show that $p_1 \geq p_2 > 0$ due to the recursive manner in which matrices U_i are generated. According to the definition, $p_1 = \min_i \max_j [a_{ij}]$ and $p_2 = (1\text{-}p_1)q_2$ where $q_2 = \min_i \max_j [a^1_{ij}]$ ($A_1 = [a^1_{ij}]$). Since $A_1 = [1/(1\text{-}p_1)] [A\text{-}p_1U_1]$, the inequality becomes $\min_i \max_j [a_{ij}] \geq \min_i \max_j [a_{ij}\text{-}p_1u^1_{ij}]$ where $U_1 = [u^1_{ij}]$. But for any fixed i, j, $a_{ij} \geq a_{ij}\text{-}p_1u^1_{ij}$ (the elements of matrix U_1 are either 1 or 0 and p_1 is positive); hence $\max_j [a_{ij}] \geq \max_j [a_{ij}\text{-}p_1u^1_{ij}]$ for any fixed i and $\min_i \max_j [a_{ij}] \geq \min_i \max_j [a_{ij}\text{-}p_1u^1_{ij}]$ thus concluding our proof. ■

Let A be a stochastic matrix whose rows are probabilistic distribution vectors $p(s_i,x)$ and which can be expressed in the form $A = \sum p_i U_i$ ($i = 1, 2,..., t < m (n\text{-}1)$) according to the above result. That means that either A has been decomposed exactly using t matrices U_i, or considering only the first t matrices in the decomposition has been satisfactory for a given level of accuracy.

Let $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_t\}$ be an auxiliary alphabet with t symbols, one for each matrix U_i in the expansion of A, and let P be a single information source over Σ emitting each σ_i with probability p_i. We give in Fig.1 a simplified block diagram of the network that synthesizes such a machine.

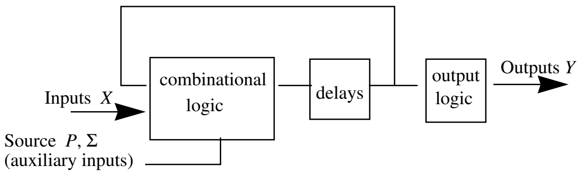


Fig.1

The combinational logic is constructed such that its output is s_j for input (x_l, σ_m, s_k) if and only if the entry of matrix U_m in the row corresponding to (s_k, x_l) and the column corresponding to s_j equals 1. We refer to $log_2|\Sigma|$ inputs needed to encode the source as *auxiliary inputs*. The output logic box is a combinational logic implementing the function Λ.

<u>Interpretation:</u> Theorem 1 states in fact that any SSM can be decomposed into a finite number of deterministic sequential machines. The behavior of the SSM is thus "simulated" by selecting one of these deterministic machines based on the values of the auxiliary inputs.

<u>Example:</u> Let's synthesize now the stochastic machine $\mathcal{M}$ defined by $\mathcal{M} = (S, X, Y, \{A(x)\}, \Lambda)$ with $S = \{0, 1\} = X = Y$, $\Lambda(0) = 1$, $\Lambda(1) = 0$, and the following transition matrices: $A(0) = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & \frac{3}{4} \end{bmatrix}^T$ $A(1) = \begin{bmatrix} \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} \end{bmatrix}^T$ (using the transpose notation). Putting together $A(0)$ and $A(1)$ we have the whole transition matrix A as: $A = \begin{bmatrix} A(0) \\ A(1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{2}{3} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{4} & \frac{1}{3} & \frac{1}{2} \end{bmatrix}^T$.

Applying Theorem 1, we get

$$A = \frac{1}{2}\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}^T + \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}^T + \frac{1}{6}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}^T + \frac{1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}^T$$

thus $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ and $P = (1/12, 1/4, 1/6, 1/12)$. Encoding the symbols in Σ with 2 bits (w_1, w_2) as 00, 01, 10, 11 respectively, we get the following transition table.

w_1 w_2	x	$s_1^{(n)}$	$s_1^{(n+1)}$	y
0 0	0	0	0	1
0 0	0	1	1	0
0 0	1	0	0	1
0 0	1	1	0	0
0 1	0	0	1	1
0 1	0	1	0	0
0 1	1	0	1	1
0 1	1	1	1	0
1 0	0	0	1	1
1 0	0	1	1	0
1 0	1	0	0	1
1 0	1	1	1	0
1 1	0	0	1	1
1 1	0	1	1	0
1 1	1	0	1	1
1 1	1	1	1	0

Using a standard synthesis approach, we obtain a circuit which synthesizes this SSM, as shown in Fig.2 (words Σ on bits (w_1, w_2) must be supplied with the probability distribution P).

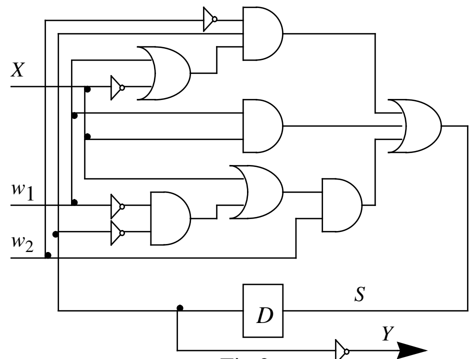


Fig.2

In summary, the basic synthesis procedure based on Theorem 1, involves essentially the synthesis of a combinational circuit with feedback and the construction of information sources with prescribed probability distributions.

III. Constrained Sequence Generation

In this section, we give a precise characterization of sequences in terms of their transition matrices. We provide also an exact formulation of the constrained sequence generation problem and propose a block-oriented approximation algorithm for solving it.

A. Sequence characterization

In what follows, we associate with every Moore-type SSM its output sequence (of length $L \geq 1$), generated during its normal operation, and we will interchangeably refer to both SSM and its output sequence. For practical purposes, we restrict our attention only to reduced stochastic machines of Moore-type [12].

<u>Definition 5.</u> (Output Equivalence)

Two reduced stochastic machines $\mathcal{M}$ and $\mathcal{M}^*$ are *output-equivalent* if the following conditions are satisfied:

1) The state spaces $S^{\mathcal{M}}$ and $S^{\mathcal{M}^*}$ have the same cardinality, that is, $|S^{\mathcal{M}}| = |S^{\mathcal{M}^*}|$;

2) The output spaces $Y^{\mathcal{M}}$ and $Y^{\mathcal{M}^*}$ are the same, that is $Y^{\mathcal{M}} = Y^{\mathcal{M}^*}$;

3) For every state s_i of $\mathcal{M}$ there corresponds a state s_j of $\mathcal{M}^*$, and vice versa, such that $\Lambda^{\mathcal{M}}(s_i) = \Lambda^{\mathcal{M}^*}(s_j)$ for every input u with $L(u) \geq 1$.

<u>Interpretation:</u> If the isomorphism relationship between state spaces S^M and S^{M^*} is given by the function $h : S^M \rightarrow S^{M^*}$, then we can represent the output-equivalence relationship between machines M and M^* as in Fig.3:

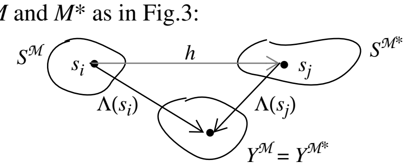


Fig.3

Basically, $S^{\mathcal{M}}$ is isomorphically mapped to $S^{\mathcal{M}^*}$ such that the output spaces coincide. These considerations translate into a definition for sequence equivalence as follows:

<u>Definition 6.</u> (Sequence Equivalence)

The output sequence $Y^{\mathcal{M}}$ generated by machine $\mathcal{M}$ is ε-*equivalent* with the output sequence $Y^{\mathcal{M}^*}$ produced by $\mathcal{M}^*$ if $\|A^{\mathcal{M}} - A^{\mathcal{M}^*}\| < \varepsilon$, where the norm is defined as $\|A\| = max|a_{ij}|$. (We note the special case $\varepsilon = 0$ when $A^{\mathcal{M}} = A^{\mathcal{M}^*}$, which corresponds to exact equivalence).

Differently stated, two output sequences are ε-equivalent if they are generated by SSMs characterized by nearly the same average transition probabilities, that is $\left|a_{ij}^{\mathcal{M}}(x) - a_{ij}^{\mathcal{M}^*}(x)\right| < \varepsilon$, for any input x.

B. The generation procedure

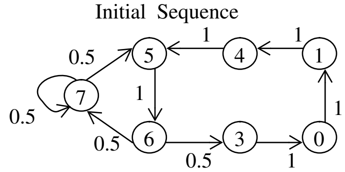
In practice, we want to generate a fixed-length sequence satisfying a certain set of constraints or, more frequently, we may have obtained from simulation a characteristic sequence for a target circuit and we want to compact it into a new one while preserving its statistics. The first situation was illustrated in Section II.B. Let's analyze now the second case by considering the problem of synthesizing a SSM from a given vector sequence.

Example: Assume that the following short sequence of 20 input vectors ($v_1, v_2, ..., v_{20}$) is representative of the input data for some target circuit.

$v_{20} \cdots$ $\quad\quad\quad\quad\quad\quad\quad\quad$ $\cdots v_2\ v_1$

0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0
0 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0
0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1

Initial Sequence



Transition Graph

Fig.4

In the lower part of Fig.4, we have the transition graph corresponding to this sequence. The 'state' nodes are labelled with the values that appear in the initial sequence (decimally encoded), while the labels on the edges are conditional probabilities captured by analyzing the initial sequence. For instance, the word '111'is half of the time followed by '101' and the other half by itself, thus we have $a_{75} = 0.5$ and $a_{77} = 0.5$.

Let $\mathcal{M}$ be the SSM associated with this sequence; as we can see, $S^{\mathcal{M}} = \{0, 1, 3, 4, 5, 6, 7\}$ and $|S^{\mathcal{M}}| = 7$. We are trying to synthesize a new machine $\mathcal{M}^*$, output-equivalent with $\mathcal{M}$, and eventually to generate an equivalent (and compacted) sequence with the initial one, using $\mathcal{M}^*$. To make our job easier, let's assume also that $Y^{\mathcal{M}} = S^{\mathcal{M}}$ and $Y^{\mathcal{M}^*} = S^{\mathcal{M}^*}$. From the very beginning, just by looking at the first two conditions in Definition 5, we may deduce that $|S^{\mathcal{M}^*}| = 7$ and $Y^{\mathcal{M}^*} = Y^{\mathcal{M}} = S^{\mathcal{M}} = S^{\mathcal{M}^*}$. The corresponding stochastic matrix A for the initial sequence is shown below, along with its decomposition $A^{\mathcal{M}} = \frac{1}{2} \cdot U_1 + \frac{1}{2} \cdot U_2$, according to Theorem 1:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} + \frac{1}{2} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence, we need a single auxiliary bit w to distinguish between the two deterministic sequential machines obtained: $w = 0$ specifies the first machine which corresponds to U_1 and $w = 1$ the second machine (both with probability 0.5) which corresponds to U_2. An implementation of $\mathcal{M}^*$ (using D Flip-Flops) is given in [8]. This SSM can now be used as a generator for a 3-bit sequence with the same stochastic characteristics as the original one. The bit w is generated using a random number generator such that 0 and 1 are equally likely (i.e. probability 0.5). To generate a sequence, the SSM $\mathcal{M}^*$ should be initialized in the most probable state: in our case, either '110', '101' or '111' can be used. Using a random number generator for the bit w, we get the following behavior when considering '111' as the initial state.

step	w	y_1^n	y_2^n	y_3^n	y_1^{n+1}	y_2^{n+1}	y_3^{n+1}
1	0	1	1	1	1	0	1
2	1	1	0	1	1	1	0
3	1	1	1	0	1	1	1
4	1	1	1	1	1	1	1
5	0	1	1	1	1	0	1
6	1	1	0	1	1	1	0
7	0	1	1	0	0	1	1
8	0	0	1	1	0	0	0
9	0	0	0	0	0	0	1
10	1	0	0	1	1	0	0

Analyzing the next state bit lines, we get the following stochastic matrix after 10 generated vectors:

$$A^{\mathcal{M}^*} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}$$

Note: we can see that for 10 generated vectors with $\mathcal{M}^*$, the initial stochastic characteristics are preserved exactly; indeed from Definition 6 with $\varepsilon = 0$, we have $\left\|A^{\mathcal{M}} - A^{\mathcal{M}^*}\right\| = 0$, therefore, in this case, a compaction ratio of 2 has been achieved without any loss of information.

In practice, we may have to deal with sequences that have a large number of bits (and bit patterns) which will give rise to large number of states in the SSM. The manipulation of matrix A thus becomes prohibitive. To handle such cases, we apply the above procedure in a *block-oriented* fashion, that is first partition the whole sequence of say n bits, into b smaller groups of at most $\left\lfloor \frac{n}{b} \right\rfloor$ bits, and after that apply the procedure to each block one at a time. By doing so, some accuracy is sacrificed by ignoring dependencies across block boundaries, but the ability to work with sequences having a large number of bits is significantly increased.

We decide whether or not a set of bits are in the same block by considering only correlations between pairs of bits. More formally, given a set of bits $\{x_i\}_{1 \le i \le n}$ to be partitioned into b groups $G_1, G_2, ..., G_b$, we construct a complete graph on n vertices where each vertex corresponds to one of the bits and each edge corresponds to the pairwise correlation between the corresponding bits. The edge weights are defined by $cost(x, y) = \sum_{i, j, k, l = 0, 1} \left|p(x_{i \to j} y_{k \to l}) - p(x_{i \to j}) \cdot p(y_{k \to l})\right|$ for edge (x, y) in the *bit-dependency graph*. Next, we find an

Table 1: Total power (uW@20MHz) for S_1

Circ.	#Inp.	Exact Power	r = 50, $k = 4$	r = 50, $k = 6$	r = 100, $k = 4$	r = 100, $k = 6$
C1355	41	4218.000	4251.00	4232.40	4276.80	4236.00
C1908	33	6969.00	6970.80	7019.40	6894.00	6966.60
C3540	50	19603.20	19654.20	19393.80	19497.00	19102.20
C432	36	3070.80	3054.00	3018.00	3065.40	3024.60
C499	41	5374.820	5390.40	5374.80	5431.20	5397.00
C6288	32	347886.00	351669.60	348599.40	354933.60	349987.20
C880	60	5990.40	6018.60	6021.60	6084.00	5976.60
s1196	14	7698.60	7492.20	7512.60	7594.20	7452.60
s344	9	1814.40	1841.40	1849.20	1851.60	1865.40
s641	35	2908.80	2779.80	2798.40	2805.00	2806.20
s838	34	1551.00	1538.40	1540.20	1554.60	1503.60
s9234	36	21693.60	21081.60	21081.60	21673.20	21042.6
	Average Error (%)		1.13	1.33	1.05	1.81
s298	3	975.00	974.40		972.60	
s386	7	1996.80	2022.60		2023.20	
	Average Error (%)		0.68		0.78	

Table 2: Total power (uW@20MHz) for S_2

Circ.	Exact Power	$r = 5$	$r = 10$
C1355	3783.17	3863.27	3918.51
C1908	6352.03	6683.00	6592.43
C3540	14471.32	12603.73	13034.91
C432	1809.95	1706.08	1860.58
C499	4390.45	4470.10	4467.74
C6288	104117.45	95628.77	92198.86
C880	3787.93	3526.17	3716.96
	Average Error (%)	6.11	5.06

assignment of each vertex to some group such that the expression

$$\sum_{1 \le p < q \le k} \sum_{x \in G_p, y \in G_q} \alpha(x, y) \cdot cost(x, y)$$

is minimized (where $0 \le \alpha(x,y) \le 1$ is a weighting coefficient that represent the topological distance between the x and y inputs in the circuit involved in the compaction process). This is in fact a *multi-way partitioning* problem which is NP-complete in the general case [13]. Fortunately, excellent heuristics are available to solve this problem [14][15].

IV. Experimental Results

The problem of sequence compaction is related to that of sequence generation. Because the latter is contained as a step in the compaction process, we will address the generation problem through the compaction problem.

In all experiments, we target *lossy compression* [16], 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 power consumption is concerned. If the initial sequence has the length L_0 and it turns out that the new generated sequence is of length $L < L_0$, then the outcome of this process is a compacted sequence equivalent to the initial one as far as total power consumption is concerned; we say that a *compaction ratio* of $r = L_0/L$ was achieved.

We assume that the input data is given in the form of a sequence of binary vectors. The global strategy is depicted in Fig.5 and follows the steps described in Section III.

Starting with an n-bit input sequence of length L_0, we extract the initial set of statistics and based on it, if the number of bits n is too large to be managed as a whole, the set of input bits is partitioned into b subsets (blocks) according to the Kernighan-Lin heuristic proposed in Section III. To every block, we associate a stochastic machine (SSM_1, SSM_2,...,SSM_b in Fig.6b). This is similar to approximating a single source on a large number of bits with many independent sources, each one having a smaller number of bits.

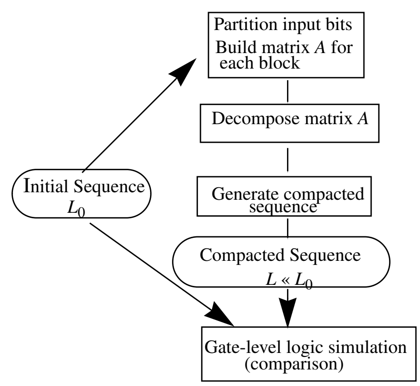
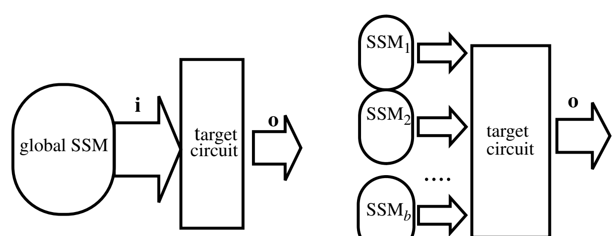


Fig.5

We note that, as a side effect, this strategy may introduce new vectors in the final compacted sequence that were absent in the original sequence.



(a) (b)

Fig.6

Once a partitioning solution is obtained, we apply the algorithm in Section III to each group of bits. From our experience, this strategy works well (less than 5% relative error on average) for pseudorandom or moderately biased input sequences. If the sequence to be compressed is a highly correlated one, then this approach will rise the level of the error about 10-15% on average. Therefore, in such cases, the global modeling of the SSM (as depicted in Fig.6a) is preferred.

Finally, a validation step is included in the strategy; using an in-house gate-level logic simulator (which does account for spurious activity in the circuits) developed under SIS, the total power consumption of some ISCAS benchmarks has been measured for the initial and the compacted sequences, making it possible to assess the effectiveness of the compaction procedure (under both zero- and real-delay models).

In Tables 1-2, we provide only the real-delay gate-level simulation results for two initial sequences: S_1 of length L_0=100,000, which is a moderately biased sequence and S_2 of length 4,000, which is a highly biased sequence taken from industry. As shown in Table 1, S_1 was compacted with two different compaction ratios (namely r = 50 and r = 100) using the strategy in Fig.6b. For each value of the compaction ratio, different sizes were allowed for the number of bits per block (k = 4, 6). For two of the sequential circuits (s298 and s386) the partitioning step was unnecessary due to the small number of input bits (3 and 7, respectively).

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

On the other side, S_2 was compacted using the strategy illustrated in Fig.6a for two compaction ratios (r = 5 and r =10). As reported in Table 2, the results are still good, the average relative error being below 10% on average.

On the efficiency side, the running times obtained for ε = 0.01 (cf. definition 6) vary from a few to tens of CPU seconds on a Sun SPARC 20. Setting ε = 0.1 will reduce the running time by one order of magnitude; in this way one can trade-off accuracy vs. run-time (as guaranteed by Theorem 2) when this is satisfactory from a practical point of view.

As an important observation, we note that 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. In such cases, the decomposition becomes the critical step as far as running time is concerned.

As the results demonstrate, large compaction ratios (1-2 orders of magnitude) can be obtained in a short amount of time with a small loss in accuracy for total power prediction, either for combinational or sequential circuits. From this perspective, simulative approaches will significantly benefit from these results.

V. Conclusion

In this paper, we addressed the problem of stochastic machines synthesis targeting constrained sequence generation or compaction. Shifting the attention from the ‘circuit problem’ to the ‘input problem’, we proposed an original approach to generate input sequences (which must satisfy a set of statistics) and to compact an existing sequence into a much shorter equivalent one.

The mathematical foundation of this approach relies in probabilistic automata theory and based on this, a general procedure for SSM synthesis is revealed. After that, these machines can be used in a stand-alone mode for sequence generation or compaction.

The issues brought into attention on this paper are new to the power community and represent a first step to reduce the gap between simulative and probabilistic techniques which are currently the norm.

References

[1] 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

[2] 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. 270-275, June 1992.

[3] F. N. Najm, ‘Transition Density, A Stochastic Measure of Activity in Digital Circuits’, in Proc. ACM/IEEE Design Automation Conference, pp. 644-649, June 1991.

[4] C.-Y. Tsui, M. Pedram, and A. M. Despain, ‘Efficient Estimation of Dynamic Power Dissipation with an Application’, in Proc. IEEE/ACM Intl. Conference on Computer-Aided Design, pp. 224-228, Nov. 1993.

[5] B. Kapoor, ‘Improving the Accuracy of Circuit Activity Measurement’, in Proc. ACM/IEEE Design Automation Conference, pp. 734-739, June 1994.

[6] C.S.Ding, M. Pedram, ‘Tagged Probabilistic Simulation Provides Accurate and Efficient Power Estimates at Gate Level’, in Proc. of the Symposium on Low Power Electronics, pp.42-43, Sept.1995.

[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] D. Marculescu, R. Marculescu, and M. Pedram, ‘Vector Compaction Using Probabilistic Automata’, Technical Report, Univ. of Southern California, March 1996.

[9] 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.

[10] J. Von Neumann, ‘Probabilistic Logics and Synthesis of reliable organisms from unreliable components’, in Annals of Mathematics Studies, Vol.34, pp.43-98, Princeton Univ. Press, Princeton, New Jersey 1956.

[11] A. Davis, ‘Markov Chains as Random Input Automata’, in American Mathematical Monthly, Vol.68, pp. 264-267, 1961.

[12] M. Rabin, ‘Probabilistic Automata’, in Information and Control, Vol.6, pp. 230-245, 1963.

[13] M. Garey, and D. Johnson, ‘Computers and Intractability’, New York: Freeman, 1979

[14] B. Kernighan and S. Lin, ‘An Efficient Heuristic Procedure for Partitioning Graphs’, in Bell Systems Technical Journal, Vol.49, No.2, pp.291-307, 1970.

[15] C. Fiduccia and R. Matheyses, ‘A Linear-Time Heuristic for Improving Network Partitions’, in Proc. ACM/IEEE Design Automation Conference, pp. 175-181, June 1982.

[16] J. Storer, ‘Data Compression: Methods and Theory’, Ch.1, Computer Science Press, 1988.