

A Markov Chain Sequence Generator for Power Macromodeling

Xun Liu Marios C. Papaefthymiou

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, Michigan 48109

ABSTRACT

In this paper, we present a novel sequence generator based on a Markov chain model. Specifically, we formulate the problem of generating a sequence of vectors with given average input probability p , average transition density d , and spatial correlation s as a transition matrix computation problem, in which the matrix elements are subject to constraints derived from the specified statistics. We also give a practical heuristic that computes such a matrix and generates a sequence of l n -bit vectors in $O(nl + n^2)$ time. Derived from a strongly mixing Markov chain, our generator yields binary vector sequences with accurate statistics, high uniformity, and high randomness. Experimental results show that our sequence generator can cover more than 99% of the parameter space. Sequences of 2,000 48-bit vectors are generated in less than 0.05 seconds, with average deviations of the signal statistics p , d , and s equal to 1.6%, 1.8%, and 2.8%, respectively.

Our generator enables the detailed study of power macromodeling. Using our tool and the ISCAS-85 benchmark circuits, we have assessed the sensitivity of power dissipation to the three input statistics p , d , and s . Our investigation reveals that power is most sensitive to transition density, while only occasionally exhibiting high sensitivity to signal probability and spatial correlation. Our experiments also show that input signal imbalance can cause estimation errors as high as 100% in extreme cases, although errors are usually within 25%.

1. INTRODUCTION

The basic idea behind power macromodeling is to generate a mapping between the power dissipation of a circuit and certain statistics of its input signals. The most commonly used statistics are the average signal probability p , the average transition density d , and the spatial correlation s . Power macromodeling proceeds in two phases. In the characterization phase, the circuit is simulated under sample input streams with various signal statistics to obtain power dissipation values. This mapping is used in the evaluation phase to predict the power dissipation of the circuit based on the statistics of its actual input signals.

A key issue in power macromodeling is the fast generation of input sequences with different and diverse statistics during the characterization phase. The creation of synthetic sequences can amount to a significant portion of macromodel characterization time. More importantly, sequence generation affects the accuracy of macromodeling. Since the actual input signals are unknown during the characterization step, sequences with all possible statistics must be applied to circuit simulation. Large estimation errors may arise when the actual input signal statistics fall outside the range of the characterization statistics.

In this paper, we present a novel sequence generator based on Markov chain (MC) modeling. We first present a precise mathematical framework for the problem of generating a sequence of random vectors with a given average signal probability p , average transition density d , and spatial correlation s . Specifically, we formulate the sequence generation problem as a transition matrix com-

putation with $O(2^n)$ constraints and $O(2^{2n})$ unknowns. We show that this exponential-size formulation captures all vector sequences of arbitrary length that satisfy the given statistics. Moreover, by relying on Markov chain theory, we argue that the n -bit vector sequences generated by solving this formulation have accurate statistics, superior uniformity, and high randomness.

We subsequently move on to describe a reduced version of the original matrix computation problem with only $O(n^2)$ constraints and $O(n)$ unknowns. We argue that the resulting sequences are still characterized by high accuracy, uniformity, and randomness, just like the ones derived from the original exponential-size formulation. We then describe a practical heuristic for generating vector sequences by solving this reduced formulation. The worst-case runtime of our heuristic generator is $O(nl + n^2)$, where n is the width and l is the length of the generated vector sequence.

Experimental results confirm our theoretical analysis. Our generator is capable of producing vector sequences with statistics that cover more than 99% of the statistics space derived by mathematical analysis. The generated sequences have superior uniformity and precise statistics. For sequences of 2,000 48-bit vectors, average deviations of the signal statistics are 1.6%, 1.8%, and 2.8% for p , d , and s , respectively. The CPU time for generating each $48 \times 2,000$ sequence is less than 0.05 seconds.

Our signal generator can serve as a powerful tool for the analysis of power macromodels. To demonstrate its high utility, we have studied the effect of each macromodel parameter on power dissipation. By computing the sensitivity of power dissipation to the three statistics p , d , and s for ISCAS85 benchmark circuits, we have found that transition density is the most power affecting parameter, while signal probability and spatial correlation show large power sensitivities only occasionally. Our investigation has also revealed that power macromodels statistics are in general robust to imbalances of their input signal statistics, although high estimation errors may still arise in extreme cases.

The remainder of the paper has 7 sections. In Section 2, we give background on power macromodeling and sequence generation. In Section 3, we present the precise mathematical formulation of our MC-based sequence generator. In Section 4, we describe a clustering scheme to derive a low-complexity formulation while retaining all the desirable properties of the original formulation. Our heuristic for solving the reduced problem formulation is given in Section 5. Our sequence generator is evaluated in Section 6. Section 7 describes the application of our generator to the study of power macromodeling. Section 8 summarizes our contributions.

2. BACKGROUND

2.1 Power Macromodeling Characterization

A power macromodel is a mapping between the power dissipation of a circuit under a specific input sequence and certain statistics of that sequence. Various mapping methods have been proposed in the literature. A look-up table (LUT) approach is introduced in [15] and improved in [4]. The LUT stores the estimates for equally

spaced discrete values of the input signal statistics. Interpolation is used for the estimates of the input statistics not in the LUT. Zhanping *et al* introduce the concept of power sensitivity which can be used to analyze and improve the accuracy of interpolation [8, 9]. Analytical power macromodeling uses mathematical expressions to calculate estimates and, therefore, avoids the space cost [5, 16]. The commonly used expressions are low-order polynomial functions.

Besides mapping methods, different statistics have been chosen to build power macromodels. Input signal entropy was proposed in [20] and applied to power macromodeling in [13]. Temporal correlation was introduced in [12] and applied to power macromodeling in [5]. In this paper, we choose average input signal probability p [4, 15, 18], average input transition density d [4, 15], and input spatial correlation s [4, 15] as the input parameters to generate the macromodels. Given a circuit block with n inputs and a binary input stream $I = \{(i_{11}, i_{12}, \dots, i_{1n}), \dots, (i_{l1}, i_{l2}, \dots, i_{ln})\}$ of length l , these metrics are defined as follows:

$$p = \frac{\sum_{j=1}^n \sum_{k=1}^l i_{kj}}{n \times l} \quad (1)$$

$$d = \frac{\sum_{j=1}^n \sum_{k=1}^{l-1} i_{kj} \oplus i_{k+1j}}{n \times (l-1)} \quad (2)$$

$$s = \frac{\sum_{j=1}^n \sum_{k=1, k \neq j}^n \sum_{l=1}^l i_{nk} \oplus i_{nj}}{l \times \lceil n/2 \rceil \times \lfloor n/2 \rfloor} \quad (3)$$

Once the input parameters are chosen, sample input streams with different p , d , and s are generated. The power dissipation \mathcal{P} of a given circuit is then derived by simulations with these input signals to create the macromodel

$$\mathcal{P} = g(p, d, s), \quad (4)$$

where g is a mapping procedure. In the evaluation phase, the p, d, s of the actual input sequence are computed and the power dissipation is predicted using g . In case the actual p, d , and s do not match any of those used in the characterization, approximation methods such as interpolation are applied.

The number of different sequences used in the characterization phase and the distribution of their signal statistics significantly affect the quality of g and eventually the power estimation result. The statistics of the characterization sequences need to cover the entire space of the possible statistics. Furthermore, the distribution of these statistics must be carefully chosen. *Power sensitivity* to an input metric K can be used to show how K affects \mathcal{P} and is defined as:

$$\lim_{\Delta K \rightarrow 0} \frac{\Delta \mathcal{P}}{\Delta K} \quad (5)$$

Intuitively, the higher the sensitivity of K is, the more different points of K should be used in the characterization phase to increase the accuracy of power macromodeling.

2.2 Sequence Generation

Sequence generation has been widely used in many fields such as coding, simulation, cryptography, and verification. The related literature is very extensive, and this section can only give a very brief and certainly non-exhaustive discussion.

Research on sequence generation can be divided into several different areas. One of the very active fields is pseudo-random number generation (PRNG). The objective of PRNG is to generate a sequence of numbers with uniform distribution within a given set [3, 6, 23, 24]. The most frequently used PRNG algorithm is the linear congruential random number generation method [17]. In this

approach, starting from an initial number X_0 , called the *seed*, a sequence of binary numbers X_k is generated by

$$X_k = (AX_{k-1} + B) \bmod M, \quad (6)$$

where A , B , and M are parameters. The results from PRNG can be modulated to create random vector sequences with certain probability distribution functions. On the other end of the spectrum, deterministic sequence generation (DSG) tries to produce specific signal patterns. One of the most important DSG applications is circuit testing [1, 2]. (Sequence generation in random testing [26] can be considered as a special case of DSG, because its objective is to produce all patterns exhaustively and avoid any repetition.)

Sequence generation for power macromodeling (SGPM), which is the problem addressed in this paper, is different from both PRNG and DSG. Unlike PRNG, the required sequences must satisfy certain statistical properties, i.e. the given signal probability, transition density, and spatial correlation, and do not necessarily follow the uniform distribution. In addition, the sequences must exhibit high randomness and do not need contain specific patterns or avoid repetition as in DSG. SGPM also differs from sequence synthesis/compaction (SSC) techniques [7, 11, 19, 22]. In SSC, a sequence is derived from a given longer sequence. The objective of SSC is to ensure that the derived sequence maintains the same properties, e.g. power effects, as the original one. Consequently, the shorter sequence can replace the longer one in simulations to reduce computation time. SGPM is actually the procedure to generate the original long sequences. It is complementary to SSC, which can be the post-processing step of SGPM to further speedup macromodel characterization.

Previous power macromodeling research has lacked an efficient and mathematically sound way for generating input sequences with given signal statistics. There is little analysis on the uniformity and statistics coverage of the generated sequences. In [5], a large number of sequences was generated based solely on average probability p . The generated sequences were inspected for other relevant statistics, and only sequences whose statistics were closest to the required ones were selected. Therefore, this approach is not guaranteed to cover the entire space of statistics, nor it controls the uniformity of the sequence statistics distribution. In [14], input sequences were generated randomly, and a local tuning method was used to make them conform to the required statistics. This approach may result in substantial non-uniformity of the sequence statistics, however, as evidenced by the data in [14]. An approach for generating sequences with a general set of given statistics using stochastic sequential machines was proposed in [21]. Due to the exponential complexity of that approach with respect to sequence width, a heuristic partitioning scheme was proposed for generating wide sequences by combining several narrow ones. No empirical evidence was presented regarding the accuracy and uniformity of the generated sequence statistics, however.

3. MC-BASED SEQUENCE GENERATION

In this section, we present the mathematical formulation of our MC-based sequence generator. Specifically, we show that sequences can be generated from a *state transition graph* (STG), which is uniquely represented by its transition matrix T . We then give a set of $O(2^n)$ constraints that relate the elements of the transition matrix with a given set of target statistics. We prove that if the transition matrix T satisfies these constraints, it is guaranteed to yield sequences with the given statistics. Furthermore, based on the structural properties of our STG, we argue that the generated sequences achieve high statistics accuracy, high uniformity, and high randomness.

3.1 Preliminaries on Markov Chains

A MC is a sequence of random variables x_i , $i = 0, 1, \dots$, with the property that the probability of x_i depends only on the value of x_{i-1} . Mathematically, this property can be represented by

$$P(x_i|x_{i-1}, x_{i-2}, \dots, x_0) = P(x_i|x_{i-1}), \quad (7)$$

where P denotes conditional probability density functions. When x_i takes only finite discrete values, a Markov sequence can be generated by constructing a STG, in which every distinct discrete value is represented by a vertex. Directed edges are introduced between every pair of vertices (including each vertex to itself) and represent state transitions. A non-negative number is assigned to each edge, representing the probability of the corresponding transition. Consequently, the STG can be uniquely defined by a transition probability matrix T , where $t_{k,j}$ equals the probability of the transition from state j to state k . Figure 1 shows a STG with two states that can be used to generate binary sequences. Its transition probability matrix is

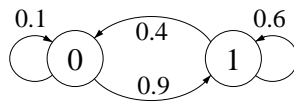
$$T = \begin{bmatrix} 0.1 & 0.4 \\ 0.9 & 0.6 \end{bmatrix}. \quad (8)$$


Figure 1: STG for a binary Markov sequence

To generate a Markov sequence from a STG, the STG is traversed starting from an arbitrary state and based on the transition probabilities. Each time a vertex is visited, its value is appended to the sequence.

If every element in the transition matrix T is strictly positive, the STG represented by T belongs to a special type of Markov chains called *strongly mixing Markov chains* (SMMCs). The sequence created from SMMCs is ergodic with uniform stationary distribution. The probability that the STG is in state k at certain time t during the traversal will converge to τ_k as t goes to infinity, no matter what the starting state is. The parameters τ_k are called *stable state probabilities* and satisfy the fixed-point equation

$$T \cdot \vec{\tau} = \vec{\tau} \quad (9)$$

and the natural constraint $\sum_k \tau_k = 1$.

3.2 Mathematical Formulation

For the remainder, we denote by $OC(b)$ the number of 1s in the binary vector representing an integer b . Accordingly, the probability $p(b)$ and the spatial correlation $s(b)$ can be computed by

$$p(b) = \frac{OC(b)}{n}, \quad (10)$$

$$s(b) = \frac{OC(b) \cdot (n - OC(b))}{\lfloor n/2 \rfloor \cdot \lfloor n/2 \rfloor}, \quad (11)$$

where n is the vector width.

The following theorem casts the problem of generating a sequence with a given p , d , and s into the problem of computing a matrix whose elements satisfy a set of $O(2^n)$ constraints.

THEOREM 1. *A sequence of n -bit binary vectors with average probability p_0 , average transition density d_0 , and spatial correlation s_0 can be generated from a STG with a $2^n \times 2^n$ transition matrix T , whose elements $t_{k,j}$ are all strictly positive and satisfy the following constraints.*

$$\exists \vec{\tau} = (\tau_0, \tau_1, \dots, \tau_{2^n-1}) : 0 < \tau_k < 1, k \in [0, \dots, 2^n - 1],$$

$$T \cdot \vec{\tau} = \vec{\tau}, \quad (12)$$

$$\vec{1}_{2^n} \cdot T = \vec{1}_{2^n}, \quad (13)$$

$$\vec{1}_{2^n} \cdot \vec{\tau} = 1, \quad (14)$$

$$p_0 = \sum_{k=0}^{2^n-1} p(k) \cdot \tau_k, \quad (15)$$

$$d_0 = \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} OC(k \oplus j) \cdot t_{k,j} \cdot \tau_k, \quad (16)$$

$$s_0 = \sum_{k=0}^{2^n-1} s(k) \cdot \tau_k, \quad (17)$$

where $\vec{1}_m$ is the m -dimensional vector of 1s.

PROOF. (sketch) Equations (12) to (14) ensure that T is the transition matrix for a MC with 2^n states. Since all elements of T are required to be positive, the MC is strongly mixing. Next, we show that Equations (15), (16), and (17) ensure that the generated sequence has the correct statistics. Each binary vector can be represented as a unique integer by enforcing a fixed bit order for all vectors. Consequently, p , d , and s can be computed using the stationary distribution $\vec{\tau}$ and transition matrix T as follows:

$$p = \sum_{k=0}^{2^n-1} p(k) \cdot \tau_k, \quad (18)$$

$$d = \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} OC(k \oplus j) \cdot t_{k,j} \cdot \tau_k, \quad (19)$$

$$s = \sum_{k=0}^{2^n-1} s(k) \cdot \tau_k, \quad (20)$$

where $k, j \in [0, 1, \dots, 2^n - 1]$. \square

3.3 Properties of SMMC-Based Generator

The sequences generated by a SMMC have accurate statistics, superior uniformity, and high randomness, properties that are particularly desirable for power macromodeling characterization. The statistical parameters p , d , and s can be controlled accurately by appropriately selecting $\vec{\tau}$ and T as in Theorem 1. Furthermore, since T is fixed and the state probabilities of the STG converge to $\vec{\tau}$ after the initial warm-up period, p , d , and s also become stable, indicating that the generated sequences have high uniformity.

The sequences also have high randomness due to the Markov chain model. In Markov sequences, two non-adjacent vectors are independent. From an information theory standpoint, independence means large information entropy, which is a measure of randomness. From Equation (9), it is straightforward to prove that, if all elements of T are positive, the stationary probabilities $\vec{\tau}$ of all patterns are positive. Therefore, every pattern will appear in a sequence, provided it is long enough. The values of the stable state probabilities τ_k also affect the randomness of the generated sequences. From an entropy standpoint, for high randomness, we should select τ_k s as close to each other as possible, provided the constraints in Theorem 1 hold.

In addition to providing sequences with accurate statistics, high uniformity, and high randomness, our SMMC-based generator has a short warm-up period and is capable of producing sequences with almost all possible statistics. Specifically, we can fine tune the stable probabilities of all states and their transition probabilities based on Equations (15), (16), and (17) to achieve high coverage. We demonstrate both advantages through our empirical results in Section 6.

4. PATTERN CLUSTERING

Though the SMMC-based generator in Subsection 3.2 can generate high quality sequences, the computation of the associated transition matrix T can be a formidable task when n is large, because the number of unknowns increases exponentially with n . Specifically, we need to solve for $2^{2n} + 2^n$ unknowns to obtain T and $\vec{\tau}$.

In this section, we reduce the size of the original STG by clustering together multiple vector patterns, thus allowing to obtain sequences by solving only for the reduced transition matrix of the clustered STG. Within each cluster, we apply a specific rule to determine which state is reached. This clustering approach reduces the number of unknowns from an exponential number in the original formulation to only quadratically many. It still guarantees positive transition probabilities between any pair of states (including each state to itself), thus yielding sequences that achieve all the desirable properties of the sequences derived from the original formulation.

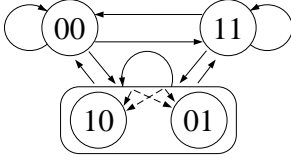


Figure 2: Clustering example

To reduce the number of variables, we group vector patterns b with the same $OC(b)$ into clusters, based on the observation that $p(b)$ and $s(b)$ solely depend on $OC(b)$ and the vector width n . The resulting clustered STG has $n + 1$ clusters. Figure 2 demonstrates the clustering procedure for $n = 2$. A total of 3 clusters is created, and the patterns within each cluster have the same number of 1s.

During sequence generation, the clustered STG is traversed in a similar manner as the original STG. When a cluster c_k is reached from a pattern b_j in cluster c_j , a 2-step method is used to decide the exact pattern being reached within c_k . In the first step, the number of bits to be inverted is determined. For a transition from c_j to c_k , the number of inversion bits x can only be in the range from $|k - j|$ to $\min(k + j, 2n - k - j)$. For each possible x , we use a function $f(x, \lambda)$ to describe the probability that x is chosen:

$$f(x, \lambda) = \frac{\lambda^{x-|k-j|} \cdot (1-\lambda)^{(\min(k+j, 2n-k-j)-x)}}{q}, \quad (21)$$

where $q = \sum_{\text{all valid } x} \lambda^{x-|k-j|} \cdot (1-\lambda)^{(\min(k+j, 2n-k-j)-x)}$ is the normalization factor, and $0 < \lambda < 1$ is a tuning parameter that allows us to control d without affecting p and s . Once the number of inversion bits x is decided, $(x + k - j)/2$ 0s and $(x - k + j)/2$ 1s in b_j are selected randomly and inverted to create the new pattern. (Note that $(x + k - j)$ and $(x - k + j)$ are always even numbers for any valid x .)

Based on the description above, we can prove that during a cluster transition $c_j \rightsquigarrow c_k$, any signal pattern in c_k can be reached with a strictly positive probability. Consequently, the STG of individual states is strongly mixing as long as the clustered STG is strongly mixing, indicating that the sequences resulting from our clustered STG maintain all the desirable properties in Subsection 3.3. (Proof is omitted due to page limitations.)

The following theorem gives the constraints for the construction of a clustered STG to generate sequences with given statistics.

THEOREM 2. *A sequence of n -bit binary vectors with average probability p_0 , average transition density d_0 , and spatial correlation s_0 can be generated by a clustered STG with an $(n + 1) \times (n + 1)$ transition matrix T , whose elements $t_{k,j}$ are all strictly positive*

and satisfy the following constraints.

$$\exists \vec{\tau} = (\tau_0, \tau_1, \dots, \tau_n) : 0 < \tau_k < 1, k \in [0, \dots, n],$$

$$T \cdot \vec{\tau} = \vec{\tau}, \quad (22)$$

$$\vec{1}_{n+1} \cdot T = \vec{1}_{n+1}, \quad (23)$$

$$\vec{1}_{n+1} \cdot \vec{\tau} = 1, \quad (24)$$

$$p_0 = \frac{1}{n} \cdot \sum_{k=0}^n k \cdot \tau_k, \quad (25)$$

$$d_0 = \sum_{k=0}^n \sum_{j=0}^n d(k, j, \lambda) \cdot t_{k,j} \cdot \tau_k, \quad (26)$$

$$s_0 = \sum_{k=0}^n s(k) \cdot \tau_k, \quad (27)$$

where $d(k, j, \lambda) = \sum_{\text{all valid } x} f(x, \lambda)$ is the average number of transition bits when moving from cluster c_k to c_j and λ is a real number in the range $0 < \lambda < 1$.

□

Theorem 2 shows that our clustering scheme reduces the number of unknowns to $n^2 + 3n + 3$ and the number of equations to $2n + 6$. Its proof is omitted due to space limitations.

5. HEURISTIC SOLUTION

Although clustering reduces the number of unknowns significantly, solving for T in Theorem 2 remains challenging due to the non-linearity of Equations (22) and (26). In this section, we present a heuristic implementation of the sequence generator. Our heuristic avoids solving non-linear equations by calculating $\vec{\tau}$ first, and then solving a linear set of constraints for T and λ . Due to the serialization of the two solution steps, our heuristic is incomplete, potentially failing to compute solutions to feasible problems. It is nevertheless highly effective in practice, as supported by our empirical results in Section 6.

5.1 Solving for the Stable Condition

In this subsection, we give a procedure for computing $\vec{\tau}$ using Equations (24), (25) and (27). This problem is under-constrained, since there are $n + 1$ unknowns and only 3 linear equations. Our procedure computes a solution that maximizes $\min\{\tau_k\}$ and therefore ensures the high randomness of the sequences generated.

Theorem 3 describes the conditions under which minimum and maximum s are achieved for a given p_0 . (Its proof is omitted due to space limitations.)

THEOREM 3. *Let \mathcal{T} be a set of tuples of dimension $n + 1$. Every tuple $\vec{\tau} = (\tau_0, \tau_1, \dots, \tau_n)$ in \mathcal{T} satisfies Equations (25) and (24). Then, the tuple with minimal s satisfies*

$$\tau_k = 0, k \in \{1, 2, \dots, n - 1\}, \quad (28)$$

and the tuple with maximal s satisfies

$$\tau_k = 0, k \in \{0, \dots, n\}, k \neq \lceil n \cdot p_0 \rceil, k \neq \lfloor n \cdot p_0 \rfloor. \quad (29)$$

□

Using Theorem 3 and Equation (27), we can compute the upper and lower bound of achievable s for a given p_0 . In the following corollary, we assume that $\lceil n \cdot p \rceil \neq \lfloor n \cdot p \rfloor$. The special case when $\lceil n \cdot p \rceil = \lfloor n \cdot p \rfloor$ can be handled by replacing $\lceil n \cdot p \rceil$ and $\lfloor n \cdot p \rfloor$ by $n \cdot p + 1$ and $n \cdot p$, respectively.

COROLLARY 4. A sequence of probability p and spatial correlation s can be created if and only if $0 \leq p \leq 1$, $0 \leq s \leq 1$, and

$$s \leq \frac{\lfloor n \cdot p \rfloor (n - \lfloor n \cdot p \rfloor) (n \cdot p - \lfloor n \cdot p \rfloor) + \lfloor n \cdot p \rfloor (n - \lfloor n \cdot p \rfloor) (\lfloor n \cdot p \rfloor - n \cdot p)}{\lfloor n/2 \rfloor \cdot \lfloor n/2 \rfloor} \quad (30)$$

□

Any probability vector $\vec{\tau}$ can be constructed by

$$\vec{\tau} = (n+1) \cdot \tau_{min} \cdot \vec{\tau}^* + (1 - (n+1) \cdot \tau_{min}) \cdot \vec{\tau}', \quad (31)$$

where $\tau_{min} = \min\{\tau_k\}$ and $\vec{\tau}^* = \frac{1}{n+1} \cdot \vec{1}$. It can be proved that Equations (24), (25), and (27) hold for $\vec{\tau}$ if and only if $\vec{\tau}'$ satisfies Equation 24 and the probability p' and spatial correlation s' of $\vec{\tau}'$ satisfy

$$p' = \frac{p_0 - ((n+1) \cdot \tau_{min}) (\frac{1}{n} \cdot \sum_{k=0}^n k \cdot \frac{1}{n+1})}{1 - (n+1) \cdot \tau_{min}}, \quad (32)$$

$$s' = \frac{s_0 - ((n+1) \cdot \tau_{min}) (\sum_{k=0}^n s(k) \cdot \frac{1}{n+1})}{1 - (n+1) \cdot \tau_{min}}. \quad (33)$$

Consequently, the problem of computing a solution $\vec{\tau}$ with maximal τ_k is converted into a problem of computing the largest τ_{min} so that p' and s' satisfy the conditions in Corollary 4.

Our procedure for computing $\vec{\tau}$ so that τ_{min} is maximized operates by performing a binary search on the possible values of τ_{min} . Initially, the lower and upper bounds of τ_{min} are set to 0 and $1/(n+1)$, respectively. In each iteration, τ_{min} is set to the mean value of the two bounds, and the values p' and s' are computed using Equations (32) and (33). If p' and s' satisfy the conditions in Corollary 4, our algorithm updates the new lower bound to be the current τ_{min} . Otherwise, the new upper bound is set to the current τ_{min} . This binary search procedure terminates when the difference between the lower and upper bounds is less than a preset threshold. It can be proved that, for the largest τ_{min} , the relation between p' and s' either satisfies Inequality (30) with strict equality or $s' = 0$. Therefore, we can compute $\vec{\tau}'$ that satisfies p' and s' based on the corresponding conditions in Theorem 3, and then construct $\vec{\tau}$ using Equation (31).

5.2 Solving for the Transition Matrix

After computing $\vec{\tau}$, we compute T and then λ such that Equations (22), (23), and (26) hold. A simple approach is to assume that the clusters are independent of each other and, therefore, $t_{k,j} = \tau_k \cdot \tau_j$. We call *independent* the matrix T_{ind} constructed under this assumption. It can be proved that the d value of the generated sequence is monotonic in λ . Therefore, we can compute the smallest and largest achievable average transition densities d_{min}^{ind} and d_{max}^{ind} , when $T = T_{ind}$, by assigning λ to 0 and 1, respectively. If $d_{min}^{ind} < d_0 < d_{max}^{ind}$, T_{ind} is our solution for T .

When $d_0 < d_{min}^{ind}$, T_{ind} cannot be used as T . We note that the unit matrix U is the correct solution for T when $d_0 = 0$, because no bit changes and there is no switch between clusters. Based on this observation and the fact that d is linear in $t_{k,j}$, we construct T as

$$T = (1 - \beta) \cdot U + \beta \cdot T_{ind}, \quad (34)$$

where $\beta = (d_0/d_{max}^{ind} + d_0/d_{min}^{ind})/2$.

When $d_0 > d_{max}^{ind}$, we use a similar approach to compute T by finding a T_{bigd} that can give the maximal transition density. Since the inversion bit number of a cluster transition $c_j \rightsquigarrow c_k$ is no more than $\min(k+j, 2n-k-j)$, the maximal possible bit inversion arises when $c_{n-k} \rightsquigarrow c_k$. Therefore, transition matrices with large $t_{k,n-k}$ can generate sequences of large d . To construct T_{bigd} , we visit every row k from top to bottom and assign the largest possible $t_{k,n-k}$.

SG(n, l, p_0, d_0, s_0)

```

1 if (Compt $\tau(n, p_0, s_0, \vec{\tau})$  == false)
2   return false
3 if (CompTand $\lambda(n, d_0, \vec{\tau}, T, \lambda)$  == false)
4   return false
5 SeqGen( $n, l, T, \lambda$ )
6 return true

```

Figure 3: Algorithm SG

The remaining elements in the same row are assigned to satisfy τ_k according to Equations (22) and (23). The algorithm finishes after every element in T_{bigd} is assigned.

The maximal achievable transition density d_{max} using our generator can be computed by setting $T = T_{bigd}$ and $\lambda = 1$. If the target transition density $d_0 > d_{max}$, no sequence can be generated. Otherwise, we can construct T using

$$T = \gamma \cdot T_{bigd} + (1 - \gamma) \cdot T_{ind}, \quad (35)$$

where $\gamma = (1 + (d_0 - d_{max}^{ind})/(d_{max} - d_{max}^{ind}))/2$.

After solving for T , we conduct a binary search to compute a value for λ so that Equation (26) holds. In all cases, every element in T is positive. Therefore, the resulting sequence generator is a SMMC system and can generate sequences with accurate statistics, high uniformity, and high randomness.

5.3 Heuristic Summary

Figure 3 gives pseudocode for our sequence generator SG. Given sequence width n , length l , and statistics (p_0, d_0, s_0) , SG returns the required sequence or reports that such a sequence does not exist. SG first computes the stationary probabilities $\vec{\tau}$ using Equations (24), (25), and (27) in lines 1–2. It then computes the clustered transition matrix and tuning parameter λ in lines 3–4 using Equations (22), (23), and (26). Finally, SG uses T and λ to generate the sequences in line 5.

The complexity of computing $\vec{\tau}$ is $O(n)$, since each of the $n+1$ elements in $\vec{\tau}$ is computed in $O(1)$ time, and the entire binary search contributes only a constant factor, given fixed accuracy. The complexity of computing T is $O(n^2)$, because each element in T_{bigd} is computed in $O(1)$ steps, and the binary search for λ takes $O(n^2)$ steps, given fixed accuracy. The complexity of generating the sequence is $O(n \cdot l)$, where n and l are the width and length of the vector sequence, respectively. It follows that the worst-case complexity of our sequence generator is $O(n^2 + n \cdot l)$.

6. PERFORMANCE EVALUATION

In this section, we give results from an empirical performance evaluation of our sequence generator. Our experiments show that our generator can yield sequences with high statistics coverage, accurate statistics, and good uniformity with quick convergence.

We implemented the proposed sequence generator using C on a SUN Ultra-Sparc II workstation with 512 MB memory. In our experiments, we generated 420 sequences with target p , d , and s evenly distributed in the 3-dimensional space. The ranges and granularities of p , d , and s were $[0.05, 0.95]$ and 0.1, respectively. (Note that sequences with some statistics combinations are impossible to generate.) Each sequence was 48 bits wide and 2,000 vectors long.

6.1 Coverage

This subsection demonstrates the capability of our generator to yield sequences with almost all possible statistics. Figure 4(a) shows the average probability p and average transition density d of our 420 sequences. The solid line represents the theoretical upper bound

of d for a given p from [16]:

$$d_{ub} = \min\{2p, 2 - 2p\}. \quad (36)$$

Figure 4(b) shows the average probability p and the spatial correlation of the same sequences. The solid line represents the theoretical upper bound of s for a given p , as given in Corollary 4. In both cases, the statistics of the generated sequences, represented by the small circles, occupy the entire space within the bounds, indicating nearly complete coverage.

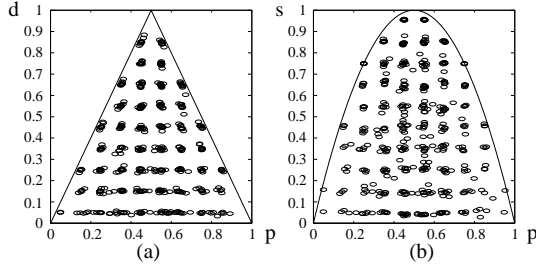


Figure 4: p,s,d distribution: (a) p-d relation (b) p-s relation

Figure 5 shows the achievable d range (as a percentage) for every (p, s) pair. It is computed by comparing the maximal achievable transition density d_{max} of our generator with the bound in Equation (36). On the average, our generator covers more than 99% of d 's range. For most (p, s) pairs, 100% coverage is achieved. The only sequences with (p, d, s) that could not be generated are in the region around $d = 1$ ($p = 0.5$). This fact reflects the trade-off between randomness and the range of statistics. (The sequences with $d = 1$ can only contain a repeating 2-vector pattern, while every vector is guaranteed to occur in our sequences.) Nevertheless, we can generate sequences that cover more than 96% of the possible d range when $p = 0.5$.

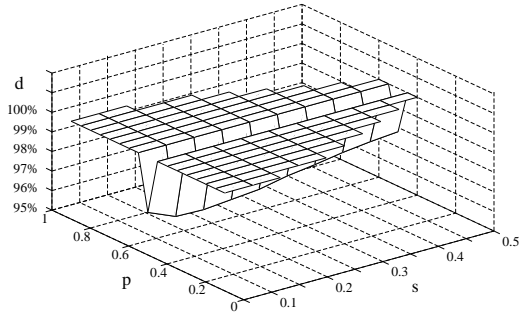


Figure 5: Coverage of d with respect to (p,s)

6.2 Accuracy

Our sequence generator can create sequences with accurate statistics. Figure 6(a) shows the average probability of the generated 420 sequences. The x-axis gives the sequence number. The target average probabilities of the sequences are also shown linked by a solid line. For almost all sequences, the average probabilities of the generated sequences overlap with their targets. Figures 6(b) and 6(c) give the similar comparisons for the transition density and spatial correlation. In both cases, most generated sequences show the exact statistics as their targets. The average relative error for p , d , and s is 1.6%, 1.8%, and 2.8%, respectively. The corresponding standard deviation is 0.017, 0.021, and 0.046.

6.3 Uniformity and Convergence

A sequence z is *uniform* if different parts of z have similar statistics. The sequences generated by our generator have superior uniformity. In our experiments, we computed the statistics of different

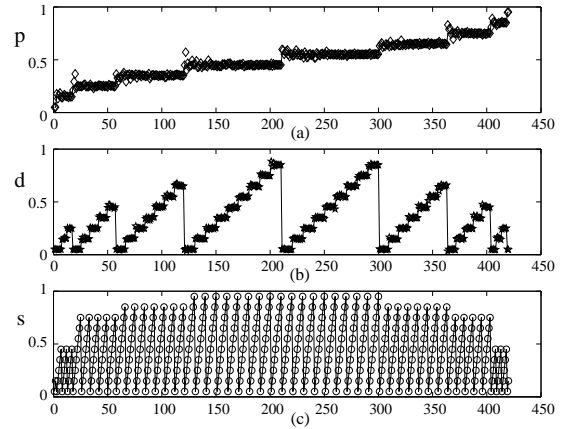


Figure 6: Accuracy of sequence statistics

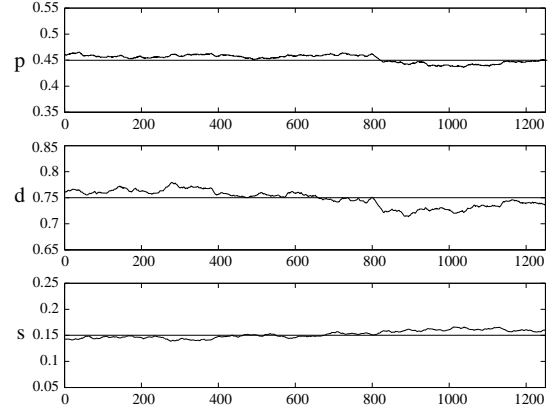


Figure 7: Statistics for different sequence fragments

fragments within each sequence. Figure 7 shows the fluctuation of p , d , and s for various fragments of a sample sequence. (Similar results were obtained from all sequences.) The x-axis gives the starting point of each fragment. All fragments have a length of 750 vectors. The target statistics are shown by the straight lines. Our graph shows that all three statistics are stable and, therefore, the sequence has good uniformity.

To analyze the convergence speed of our generator, we computed the statistics of the first k vectors of our sequences. Figure 8 shows the change of p , d , and s with respect to k for a sample sequence. The straight lines represent the target statistics. Our experiments show that warm-up length is about 750 vectors, which are computed within 0.02 seconds for a 48-bit wide sequence.

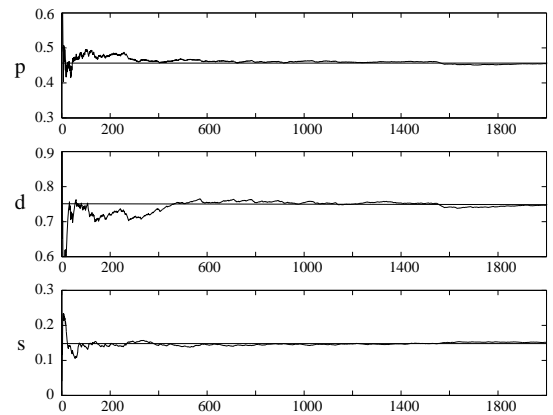


Figure 8: Statistics changes with respect to sequence length

7. POWER MACROMODELING STUDY

Our sequence generator can be applied to power macromodeling analysis, resulting in helpful insights for improving the effectiveness of power macromodels. In Subsection 7.1, we analyze the relation between various input statistics and power dissipation. Our results indicate that more accurate macromodels might be achieved by using fine granularity for d and coarse granularity for p and s in the characterization step. In Subsection 7.2, we investigate the signal imbalancing effect on the accuracy of power macromodeling. We conclude that power macromodeling is in general robust, although high estimation errors may arise in extreme cases.

7.1 Power Sensitivity Study

Our sequence generator can generate a set of sequences with the value of one statistical parameter changing over the entire parameter space while all other statistics remain fixed. Therefore, it enables us to perform extensive experiments to reveal the relation between circuit power dissipation and specific statistics of the input signals.

In our study, we designed 10 combinational circuits from the ISCAS-85 benchmark suite using a 0.35 μm standard cell library. For each circuit, we generated about 300 sequences with p , d and s evenly distributed in the 3-dimensional space. For all three parameters, the granularity was 0.1, and the range was [0.1, 0.9]. We then simulated these circuits to obtain their power dissipation using switch-level simulation with the generated inputs.

Figure 9 gives the power dissipation of circuit *c880* for input sequences with different statistics. Our results show that transition density d has the largest effect on power dissipation. The relation between d and power is nearly linear. The spatial correlation s and signal probability p do not affect power dissipation significantly, since the power surfaces in Figure 9 are quite flat. However, these two statistics do affect the power occasionally, particularly when d is small.

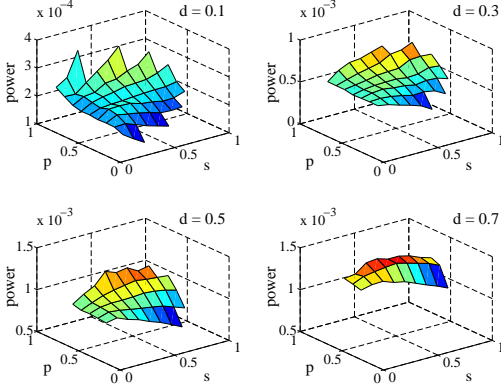


Figure 9: Power dissipation of *c880* with respect to (p, d, s)

Furthermore, we computed the average and maximal power sensitivity for all three statistics normalized by average power dissipations. Our results, given in Table 1, show that the sensitivities to p and s are much smaller than d . This observation suggests that we could use fine granularity of d and large granularity of p and s in the characterization step to achieve a more accurate macromodel and reduce model characterization time.

7.2 Input Imbalance Study

Since power macromodeling uses average signal statistics of all input bits, it implicitly assumes that all input bits have similar statistics. However, this assumption might not be true in an actual system. For example, the most significant bit in the input of a datapath component might switch less frequently than the least significant

Table 1: Power sensitivities

Design Name	P_{mean}	P_{max}	D_{mean}	D_{max}	S_{mean}	S_{max}
c1355	0.46	1.50	1.75	3.21	0.69	1.11
c1908	0.24	0.73	1.97	2.92	0.51	1.05
c2670	0.93	2.67	2.46	3.73	0.09	0.34
c3540	0.46	1.48	2.24	3.33	0.52	0.71
c432	0.76	1.96	1.80	3.49	0.81	1.98
c499	0.39	1.37	1.72	3.13	0.68	1.24
c5315	0.97	2.82	2.44	3.44	0.28	0.63
c6288	0.60	1.71	1.99	3.24	1.14	2.16
c7552	0.22	0.86	2.28	3.13	0.35	0.73
c880	1.16	3.11	2.39	4.15	0.25	0.91
average	0.62	1.82	2.10	3.38	0.53	1.09

bit. We call this phenomenon *input imbalance*. In this subsection, we analyze the effect of input imbalance on the accuracy of power macromodeling.

We applied the following investigation procedure. First, we simulated each circuit using a balanced input sequence with (p, d, s) . We then used our generator to create a set of imbalanced sequences with the same statistics and applied them to the circuit. Finally, the power dissipation of the circuits under imbalanced inputs was compared with that of the balanced sequence.

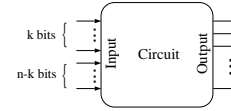


Figure 10: Signal imbalance investigation

Figure 10 illustrates our procedure for generating imbalanced sequences. For a benchmark circuit with n inputs, we randomly chose k bits and permanently set them to zero. We supplied the remaining $(n-k)$ bits with a sequence of statistics $(p_{new}(k), d_{new}(k), s_{new}(k))$. It can be proved that the statistics of the entire n -bit sequence are always equal to (p, d, s) if

$$\begin{aligned}
 p_{new}(k) &= p \cdot n / (n - k) , \\
 d_{new}(k) &= d \cdot n / (n - k) , \\
 s_{new}(k) &= (s \cdot \lceil n/2 \rceil \cdot \lfloor n/2 \rfloor - p \cdot n \cdot k) / (\lceil (n-k)/2 \rceil \cdot \lfloor (n-k)/2 \rfloor) .
 \end{aligned}$$

We generated the entire set of imbalanced sequences by increasing k from 1 until no valid $(p_{new}(k), d_{new}(k), s_{new}(k))$ could be found.

In our study, we applied all (p, d, s) combinations in Subsection 7.1 to all circuits. Figure 11 shows the comparison results for circuit *c880*, in which the maximal power increase and decrease are given with respect to the power of corresponding balanced input sequences. The worst-case increase can be as high as 100%, indicating that the imbalanced sequences can result in quite different power dissipation. Consequently, power macromodeling techniques can result in large errors in case of imbalanced input signals.

To further assess how seriously signal imbalance can affect the accuracy of power estimation, we computed the number of times that worst-case power variation remains in the $\pm 25\%$ and $\pm 50\%$ ranges. Table 2 gives our results as a percentage of all (p, d, s) combinations applied. On the average, in 68% of the cases, the worst case power variation due to signal imbalance is within 25%. In 90% of the cases, the worst-case power change due to signal imbalance is within 50% of dissipation with balanced inputs, on the average.

It should be noted that our investigation assumes absolute zero on certain inputs. This is an extreme case. We expect the power effect of input imbalance in a real situation to be less than that of Table 2. When high estimation accuracy is needed, the input signals can be partitioned into groups and characterized separately [10].

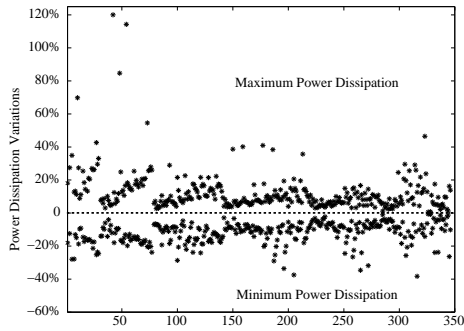


Figure 11: Power variation in c880 due to signal imbalance

Table 2: Power variations due to input imbalance.

Design Name	$\pm 25\%$	$\pm 50\%$
c1355	62%	93%
c1908	92%	95%
c2670	52%	95%
c3540	90%	93%
c432	56%	85%
c499	56%	93%
c5315	57%	84%
c6288	57%	82%
c7552	78%	84%
c880	82%	91%
average	68%	90%

8. CONCLUSION AND DISCUSSION

We have presented a framework for sequence generator based on Markov chains. Specifically, we have given a precise exponential-size formulation for the problem of generating a sequence of vectors with given average probability p , average transition density d , and spatial correlation s . We have also described a reduced formulation that retains all the desirable properties of the original one, while requiring the solution of quadratically many constraints. We have given a practical heuristic for solving the reduced formulation and generating vector sequences in $O(nl + n^2)$ time.

Our generator yields sequences with thousands of vectors in less than one second. The generated sequences achieve high coverage, accurate statistics, high uniformity, and high randomness. The proposed generator enables a detailed analysis of power macromodeling and can provide useful insights for improving its effectiveness. A public domain version of our sequence generator is available at <http://www.eecs.umich.edu/acal/software.html>.

Our sequence generator cannot provide vectors for only a very small fraction of the statistics space around $d = 1$. This fraction could be further reduced by improving our heuristic solver, e.g. by computing $\vec{\tau}$ so that each τ_k is as close to τ_{n-k} as possible, or by using an exact solver. Another interesting research topic would be the extension of our approach to generate sequences with additional statistics such as temporal correlation.

Acknowledgments

This research was supported in part by NSF under Grant No. CCR-0082876 and by ARO under Grant No. DAAD19-99-1-0304.

9. REFERENCES

- [1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. IEEE Press, Piscataway, NJ, 1995.
- [2] V. D. Agrawal and S. C. Seth. *Test Generation for VLSI Chips*. Computer Society Press, Washington D.C., 1988.
- [3] W. Aiello, S. R. Rajagopalan, and R. Venkatesan. Design of practical and provably good random number generators. *Journal of Algorithms*, 29(2):358–389, November 1998.
- [4] M. Barocci, L. Benini, A. Bogliolo, B. Ricco, and G. De Micheli. Lookup table power macro-models for behavioral library components. In *Proc. IEEE Alessandro Volta Workshop on Low Power Design*, March 1999.
- [5] G. Bernacchia and M.C. Papaefthymiou. Analytical macromodeling for high-level power estimation. In *Proc. IEEE International Conf. Computer-Aided Design*, November 1999.
- [6] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.
- [7] R. Burch, F. Najm, P. Yang, and T. Trick. A Monte-Carlo approach for power estimation. In *IEEE Trans. VLSI Systems*, pages 63–71, January 1993.
- [8] Z. Chen and K. Roy. A power macromodeling technique based on power sensitivity. In *Proc. 35th Design Automation Conf.*, June 1998.
- [9] Z. Chen, K. Roy, and T. L. Chou. Power sensitivity—a new method to estimate power dissipation considering uncertain specifications of primary inputs. In *Proc. IEEE International Conf. Computer-Aided Design*, November 1997.
- [10] R. Corgnati, E. Macii, and M. Poncino. Clustered table-based macromodels for RTL power estimation. In *Ninth Great Lakes Symp. VLSI*, pages 354–357, August 1999.
- [11] C-S. Ding, C-T. Hsieh, Q. Wu, and M. Pedram. Stratified random sampling for power estimation. In *Proc. IEEE International Conf. Computer-Aided Design*, pages 577–582, November 1996.
- [12] N. Dragone, R. Zafalon, C. Guardiani, and C. Silvano. Power invariant vector compaction based on bit clustering and temporal partitioning. In *Proc. International Symp. Low Power Electronics and Design*, August 1998.
- [13] F. Ferrandi, F. Fummi, E. Macii, M. Poncino, and D. Sciuto. Power estimation of behavioral descriptions. In *Design, Automation, and Test in Europe*, pages 762–766, March 1998.
- [14] S. Gupta. Power macromodeling for high level power estimation. *Master Thesis, Univ. of Illinois*, September 1997.
- [15] S. Gupta and F.N. Najm. Power macromodeling for high level power estimation. In *Proc. 34th Design Automation Conf.*, June 1997.
- [16] S. Gupta and F.N. Najm. Analytical model for high level power modeling of combinational and sequential circuits. In *Proc. IEEE Alessandro Volta Workshop on Low Power Design*, March 1999.
- [17] D. E. Knuth. *The Art of Computer Programming, Seminumerical Algorithms*. Addison-Wesley, 1997.
- [18] P. Landman and J. Rabaey. Activity-sensitive architectural power analysis. In *IEEE Trans. CAD*, pages 571–587, June 1996.
- [19] A. Macii, E. Macii, M. Poncino, and R. Scarsi. Stream synthesis for efficient power simulation based on spectral transforms. In *IEEE Trans. VLSI Systems*, pages 417–426, June 2001.
- [20] D. Marculescu, R. Marculescu, and M. Pedram. Information theoretic measures for power analysis. In *IEEE Trans. CAD*, pages 599–609, June 1996.
- [21] D. Marculescu, R. Marculescu, and M. Pedram. Stochastic sequence machine synthesis with application to constrained sequence generation. *ACM Trans. Design Automation of Electronic Systems*, 5(3):658–681, July 2000.
- [22] R. Marculescu, D. Marculescu, and M. Pedram. Sequence compaction for power estimation: Theory and practice. In *IEEE Trans. CAD*, pages 973–993, July 1999.
- [23] U. M. Maurer and J. L. Massey. Perfect local randomness in pseudo-random sequences. In *Advances in Cryptology—CRYPTO ’89 Proc.*, pages 100–112, August 1989.
- [24] S. Micali and C. P. Schnorr. Efficient, perfect random number generators. In *Advances in Cryptology—CRYPTO ’88 Proc.*, pages 173–198, August 1988.
- [25] A. Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Birkhauser, Boston, 1993.
- [26] V. N. Yarmolik and S. N. Demidenko. *Generation and Application of Pseudorandom Sequences for Random Testing*. John Wiley and Sons, New York, 1988.