A Markov Chain Sequence Generator for Power Macromodeling

Xun Liu Marios C. Papaefthymiou
Department of Electrical Engineering
and Computer Science
University of Michigan
Ann Arbor, Michigan 48109

Abstract

In macromodeling-based power estimation, circuit macromodels are created from simulations of synthetic input vector sequences. Fast generation of these sequences with all possible statistics is crucial for ensuring the accuracy and speed of macromodeling. 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 statistics as a transition matrix computation problem, in which the matrix elements are subject to constraints derived from the specified statistics. We also present a practical heuristic that computes such a matrix and generates a sequence of l n-bit vectors in $O(nl + n^2)$ time. Our generator is guaranteed to yield vector sequences with a given average input probability p, average transition density d, and spatial correlation s, or reports that the specified sequence type does not exist. Derived from a strongly mixing Markov chain, it generates 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 1.0%, 0.6%, and 0.6%, respectively.

Our generator enables the detailed study of power macromodeling. Using our tool and the ISCAS-85 benchmark circuits, we have assessed the power sensitivities of three input statistics. Our investigation has revealed that power is most sensitive to transition density, while only occasionally exhibiting high sensitivity to signal probability and spatial correlation. We have also investigated the power estimation error resulting from input signal imbalance. Our experiments show that signal imbalance can cause estimation error 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. Given average signal probability p, average transition density d, and spatial correlation s, our generator can produce a binary vector sequence of arbitrary length and width that satisfies these statistics, or it reports that no such sequence exists. We give a precise mathematical formulation of the sequence generation problem as a transition matrix computation with $O(2^n)$ constraints and $O(2^{2n})$ unknowns. Relying on Markov chain theory, we argue that the n-bit vector sequences generated from this exponential-size formulation have accurate statistics, superior uniformity, and high randomness. We subsequently move on to describe a practical heuristic for implementing our proposed sequence generator. Our heuristic solves a reduced version of the original matrix computation problem with only $O(n^2)$ constraints and $O(n)$ unknowns. We prove that the resulting sequences are still characterized by high accuracy, uniformity, and randomness, just like the ones derived from

the original exponential-size 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 1.0%, 0.6%, and 0.6% 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 power sensitivities of all three statistics 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. We have also investigated the power effect of signal imbalance. We conclude that power macromodels are robust in general, 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 the MC based sequence generator. In Section 4, we describe a clustering scheme to derive a low-complexity formulation with all the original desirable properties. We solve the new problem formulation 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

Power macromodeling is derived from RT-level power estimation [20, 25]. 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 [14] 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 [21] 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, 14, 18], average input transition density d [4, 14], and input spatial correlation s [4, 14] as the input parameters to generate the macromodels. Given a circuit block with n inputs and a binary input stream $I = \{(i_{11}, i_{12}, ..., i_{1n}), ..., (i_{l1}, i_{l2}, ..., i_{ln})\}$ of length l, these metrics are defined as follows [5]:

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

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

$$s = \frac{\sum_{j=1}^{n} \sum_{k=1, k\neq j}^{n} \sum_{n=1}^{l} i_{nk} \oplus i_{nj}}{l \times n \times (n-1)} \tag{3}$$

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

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

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:

$$\lim_{\Delta K \to 0} \frac{\Delta \mathcal{P}}{\Delta K} \tag{5}$$

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

$$X_k = (AX_{k-1} + B) \; mod \; M \; , \tag{6}$$

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 [27] 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. The required sequences do not follow the uniform distribution but need satisfy certain statistical properties, i.e. the given signal probability, transition density, and spatial correlation. It is impossible to generate sequences with these statistics from PRNG results, since vectors from PRNG are pseudo-independent from each other. 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. 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 was not guaranteed to cover the entire space of statistics, nor could it control the uniformity of the sequence statistics distribution. In [15], the input sequences were also generated based solely on p. A local tuning method was used to make the sequences conform to the required statistics. However, this approach could cause large variations of the statistics in different parts of the generated sequences.

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, $i = 0, 1, ...$, 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

$$P(x_i|x_{i-1}, x_{i-2}, ...x_0) = P(x_i|x_{i-1}) \; , \tag{7}$$

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

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

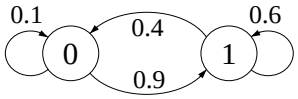

Figure 1: STG for a binary Markov sequence

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

$$T \cdot \vec{\tau} = \vec{\tau} . \tag{9}$$

3.2 Mathematical Formulation

We first introduce some notation. We denote by $OC(b)$ the number of 1s in the binary vector b. Accordingly, the probability $p(b)$ and the spatial correlation $s(b)$ can be computed by

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

$$s(b) = \frac{OC(b) \cdot (n - OC(b))}{n \cdot (n-1)} , \tag{11}$$

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.*

$$\exists \tau_k \ , \ 0 < \tau_k < 1 \ , \ k \in [0, ...2^n - 1],$$

$$T \cdot \vec{\tau} = \vec{\tau} , \tag{12}$$

$$\vec{1}_{2^n} \cdot T = \vec{1}_{2^n} , \tag{13}$$

$$p_0 = \sum_{k=0}^{2^n-1} p(k) \cdot \tau_k , \tag{14}$$

$$d_0 = \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} OC(k \oplus j) \cdot t_{k,j} \cdot \tau_k , \tag{15}$$

$$s_0 = \sum_{k=0}^{2^n-1} s(k) \cdot \tau_k , \tag{16}$$

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

Proof. (sketch) Equations (12) and (13) show that T is the transition matrix for a MC with 2^n states. Since all elements of T are positive, the MC is strongly mixing. Next, we show that Equations (14), (15), and (16) 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:

$$p = \sum_{k=0}^{2^n-1} p(k) \cdot \tau_k \ , \tag{17}$$

$$d = \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} OC(k \oplus j) \cdot t_{k,j} \cdot \tau_k \ , \tag{18}$$

$$s = \sum_{k=0}^{2^n-1} s(k) \cdot \tau_k \ , \tag{19}$$

where $k, j \in [0, ...2^n - 1]$. □

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 maximization of 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.

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. 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 and only solve for the 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.

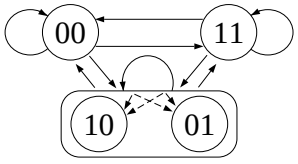

Figure 2: Clustering example

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 inverted is decided. For a transition from c_j to c_k, the *number of inversion bits* x can range only 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:

$$f(x,\lambda) = \frac{\lambda^{(x-|k-j|)} \cdot (1-\lambda)^{(\min(k+j,2n-k-j)-x)}}{q}\,, \tag{20}$$

where $q = \sum_{all\ valid\ x} \lambda^{(x-|k-j|)} \cdot (1-\lambda)^{(\min(k+j,2n-k-j)-x)}$ is the normalization factor and $0 < \lambda < 1$ is a tuning parameter. 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.

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 a $(n+1)\times(n+1)$ transition matrix T, whose elements $t_{k,j}$ are all strictly positive and satisfy the following constraints.*

$$\exists \tau_k\,, 0 < \tau_k < 1,\ \ k \in [0,\ldots,n]\,,$$

$$\begin{aligned}
T \cdot \vec{\tau} &= \vec{\tau}\,, && (21)\\
\vec{1}_{n+1} \cdot T &= \vec{1}_{n+1}, && (22)\\
p_0 &= \frac{1}{n} \cdot \sum_{k=0}^{n} k \cdot \tau_k\,, && (23)\\
d_0 &= \sum_{k=0}^{n}\sum_{j=0}^{n} d(k,j,\lambda) \cdot t_{k,j} \cdot \tau_k\,, && (24)\\
s_0 &= \sum_{k=0}^{n} s(k) \cdot \tau_k\,, && (25)
\end{aligned}$$

where $d(k,j,\lambda) = \sum_{all\ valid\ x} x \cdot f(x,\lambda)$ 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 $0 < \lambda < 1$.

□

The proof is omitted due to space limitations. Theorem 2 shows that our clustering scheme reduces the number of unknowns to $n^2 + 3n + 3$ and the number of equations to $2n + 5$.

5 Heuristic Solution

Although the clustering approach reduces the number of unknowns significantly, solving for T in Theorem 2 remains challenging due to the non-linearity of Equations (21) and (24). 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.

5.1 Solving for the Stable Condition

In this subsection, we compute $\vec{\tau}$ using Equations (23) and (25) and the following constraint derived from the basic concept of probability.

$$\vec{1}_{n+1} \cdot \vec{\tau} = 1. \tag{26}$$

We first check whether there exists a solution $\vec{\tau}$ which satisfies all three equations. Theorem 3 describes the conditions under which minimal and maximal s are achieved for a given p_0. Based on this theorem, we can compute the lower and upper bounds of s for a given p_0. (There are at most 2 unknowns and 2 linear equations in both conditions.) Then, the existence of a solution $\vec{\tau}$ for any (p_0, s_0) pair can be confirmed by comparing s_0 with these bounds.

Theorem 3 *Let $\mathcal{T}$ be a set of tuples of dimension $n+1$. Every tuple $y =< \tau_0, \tau_1, ..., \tau_n >$ in $\mathcal{T}$ satisfies Equations (23) and (26). For each y, we can calculate its s value using Equation (25). Then, the tuple with minimal s satisfies*

$$\tau_k = 0,\ k \in \{1, 2, ..., n-1\}\ , \tag{27}$$

and the tuple with maximal s satisfies

$$\tau_k = 0,\ k \in \{0, ..., n\},\ \ k \neq \lceil n \cdot p_0 \rceil,\ \ k \neq \lfloor n \cdot p_0 \rfloor\ . \tag{28}$$

□

If the solution exists, we must compute the $(n+1)$ unknowns $0 < \tau_k < 1$ using the three linear Equations (23), (25), and (26). We solve for τ_k by first generating $(n+1)$ numbers $0 < \tau_k(\theta) < 1$ from a tuning parameter $0 < \theta < 1$. The generated numbers are guaranteed to satisfy Equations (23) and (26). Furthermore, the s value of the numbers $\tau_k(\theta)$ is monotonic in θ. Thus, we can use binary search to find the θ for which the numbers $\tau_k(\theta)$ satisfy Equation (25).

5.2 Solving for the Transition Matrix

After computing $\vec{\tau}$, we compute T and then λ such that Equations (21), (22), and (24) 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 the matrix constructed under this assumption *Independent Matrix* T_{ind}. 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^{ind}_{min} and d^{ind}_{max}, when $T = T_{ind}$, by assigning λ to 0 and 1, respectively. If $d^{ind}_{min} < d_0 < d^{ind}_{max}$, T_{ind} is our solution for T.

When $d_0 < d^{ind}_{min}$, 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 can construct T as

$$T = (1-\beta) \cdot U + \beta \cdot T_{ind}\ , \tag{29}$$

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

When $d_0 > d^{ind}_{max}$, 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 $t_{k,n-k}$. The remaining elements in the same row are assigned to satisfy τ_k according to Equations (21) and (22). The algorithm finishes after every element in T_{bigd} 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

$$T = \gamma \cdot T_{bigd} + (1-\gamma) \cdot T_{ind}\ , \tag{30}$$

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

After solving for T, we conduct a binary search to compute a value for λ so that Equation (24) 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 (23), (25), and (26) in lines 1–2. It then computes the clustered transition matrix and tuning parameter λ in lines 3–4 using Equations (21), (22), and (24). 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 binary search iteration only contributes a constant factor for fixed computation accuracy. The complexity of computing T is $O(n^2)$, because each element in T_{bigd} is computed in $O(1)$ steps. 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 $O(n^2 + n \cdot l)$.

SG (n, l, p_0, d_0, s_0)
1 **if** (Compτ(n, p_0, s_0, $\vec{\tau}$) == false)
2 **return** false
3 **if** (CompTandλ(n, d_0, $\vec{\tau}$, T, λ) == false)
4 **return** false
5 SeqGen(n, l, T, λ)
6 **return** true

Figure 3: Algorithm SG

6 Performance Evaluation

In this section, the performance of the proposed sequence generator is evaluated. Experimental results demonstrate that our generator can produce 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 220 sequences with target p, d, and s evenly distributed in the 3-dimensional space. The ranges of p, d, and s were [0.05,0.95], [0.05,0.85], and [0.05,0.45], respectively. For all three parameters, the granularity was 0.1. Each sequence was 48 bits wide and 2,000 vectors long.

6.1 Coverage

This subsection demonstrates the capability of our generator to produce sequences with almost all possible statistics. Figure 4(a) shows the average probability p and average transition density d of our 220 sequences. The solid line represents the theoretical upper bound of d for a given p from [16]:

$$d_{ub} = \min(2p, 2 - 2p) . \tag{31}$$

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 derived from Theorem 3:

$$s_{ub} = \frac{2n \cdot p \cdot (1-p)}{n-1} . \tag{32}$$

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.

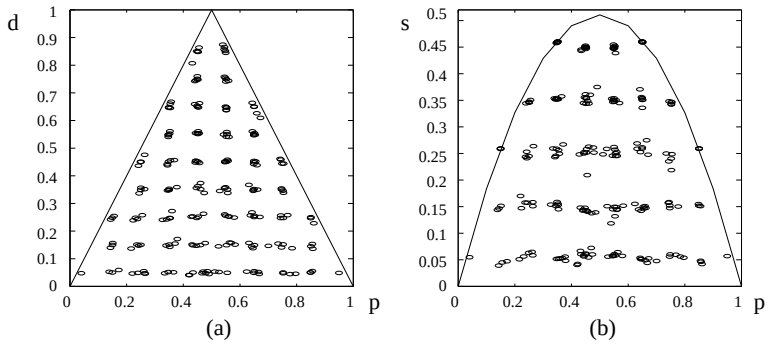

Figure 4: p,s,d distribution: (a) p-d relation (b) p-s relation

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 (31). 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 $p = 0.5$.

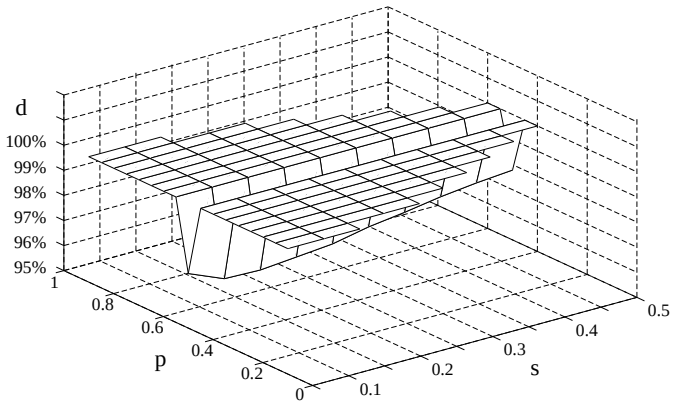

Figure 5: Coverage of d with respect to (p,s)

6.2 Accuracy

Our sequence generator can create sequences with accurate statistics. Figure 6(a) shows the average probability of the generated 220 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 1.0%, 0.6%, and 0.6%, respectively. The corresponding standard deviation is 0.014, 0.006, and 0.006.

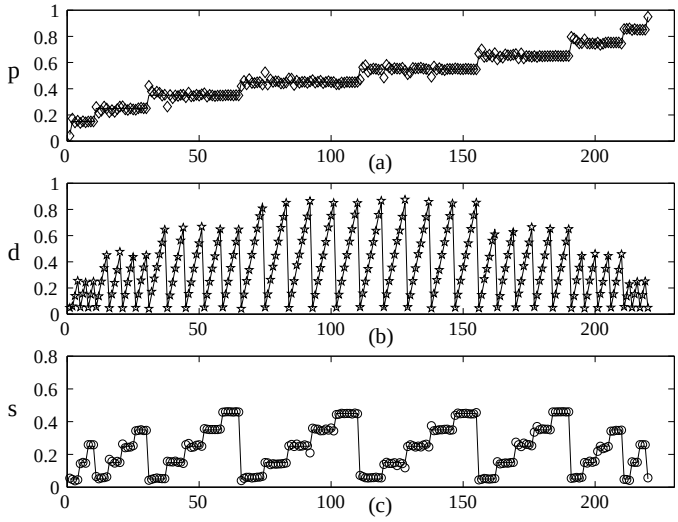

Figure 6: Accuracy of sequence statistics

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 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.

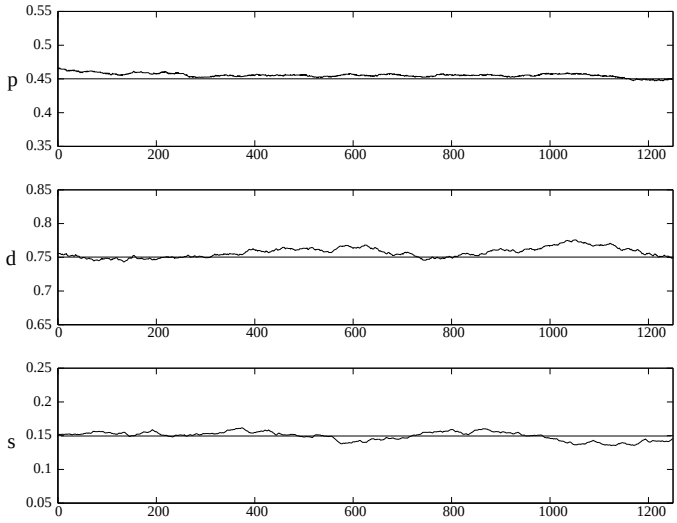

Figure 7: Statistics for different sequence fragments

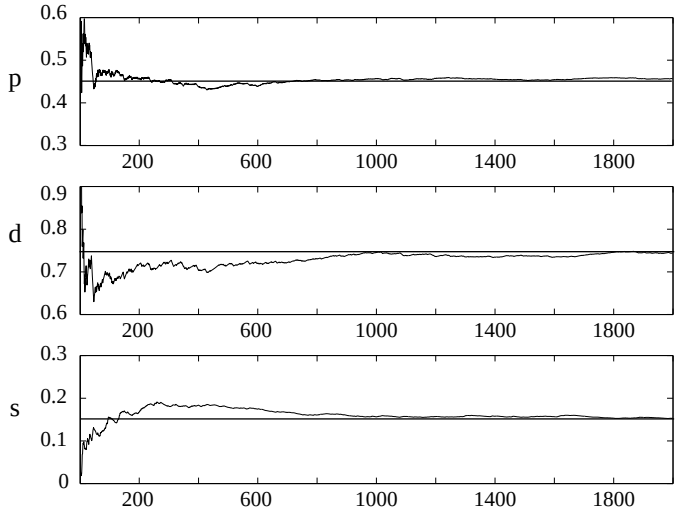

Figure 8: Statistics changes with respect to sequence length

7 Power Macromodeling Analysis

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 increasing the granularity of d and decreasing the granularity of 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 robust, although high estimation errors can 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. (Since input widths vary, the number of sequences was different for each circuit.) For all three parameters, the granularity was 0.1 and the range was [0.1,0.9]. We then simulated these circuits using switch-level simulation with the generated inputs to obtain their power dissipation.

Figure 9 gives the power dissipation of circuit $c880$ for input sequences with different statistics. Our results show that the power effect of transition density d is the largest. The relation between d and power is close to 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.

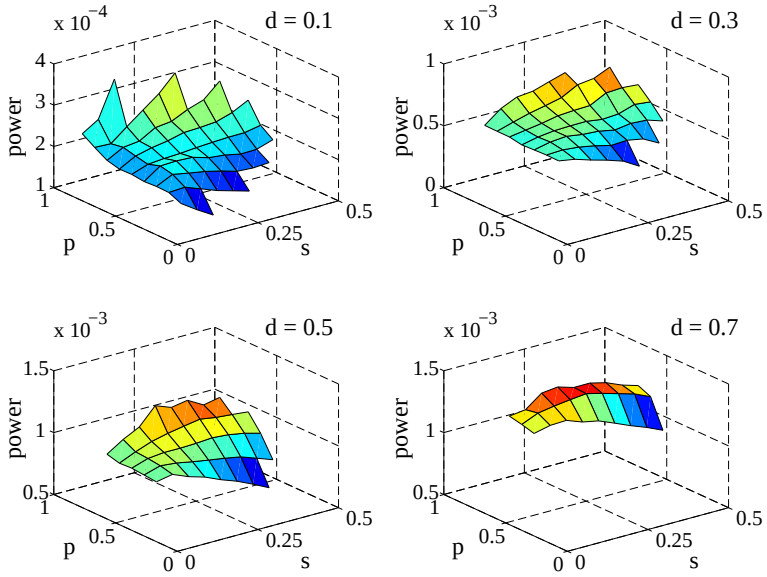

Figure 9: Power dissipation of c880 with respect to (p,d,s)

Furthermore, we computed the average and maximal power sensitivity for all three statistics normalized by average power dissipations. Table 1 gives our results. The sensitivities to p and s are much smaller than d. This observation indicates that we could increase the granularity of d and decrease the granularity of p and s in the characterization step to achieve a more accurate macromodel and reduce model characterization time.

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

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 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 dissipations of the circuits under imbalanced inputs were compared with that of the balanced sequence.

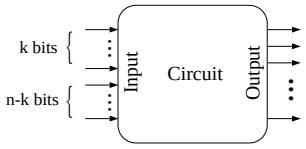

Figure 10: Signal imbalance investigation

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 (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.

We performed our study using all the (p, d, s) combinations in Subsection 7.1 for 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.

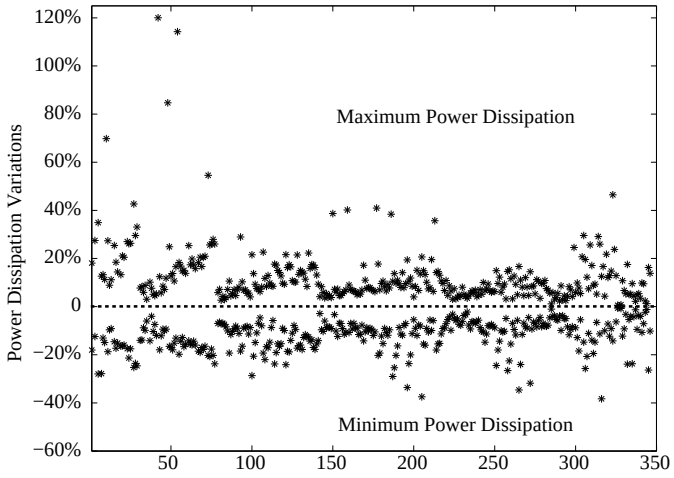

Figure 11: Power variation in c880 due to signal imbalance

To further assess how seriously signal imbalance can affect the power estimation accuracy, we have computed the percentages that the worst-case power variation remains in the $\pm 25\%$ and $\pm 50\%$ ranges. Table 2 gives our results. 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 signal 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 as in [10].

8 Conclusion

We have presented a novel vector sequence generator based on Markov chains. We have also given a practical heuristic for constructing the proposed generator and corresponding vector sequences in $O(n^2 + nl)$ time. Our generator is guaranteed to produce vector sequences with given average probability p, average transition density d, and spatial correlation s, or to report that the specified sequence type does not exist. In experiments, our generator runs extremely fast, yielding sequences covering more than 99% of the statistics space. Furthermore, the generated sequences have accurate statistics,

Table 2: Power variations due to input imbalance.

Design Name	±25%	±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%

high uniformity, and high randomness. The proposed generator enables a detailed analysis of power macromodeling and can provide useful insights for improving its effectiveness. To demonstrate the utility and power of our tool, we have used it to investigate the power sensitivities of three input statistics and to analyze the power effect of signal imbalance.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Tesing 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 *Proceedings of IEEE International Conference on 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. on 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.*, 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 *Proceedings of IEEE International Conference on 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 Symposium on 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 *Inter. Conf. on 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 *Proceedings of International Symposium on 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 and F.N. Najm. Power macromodeling for high level power estimation. In *Proc. 34th Design Automation Conf.*, 1997.

[15] S. Gupta and F.N. Najm. power macromodeling for high level power estimation. In *Univ. of Illinois Coordinated Science Laboratary Report UIUI-ENG-97-2229*, 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*. Addision-Wesley, 1997.

[18] P. Landman and J. Rabaey. Activity-sensitive architectural power analysis. In *IEEE Trans. on 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. on VLSI Systems*, pages 417–426, June 2001.

[20] E. Macii, M. Pedram, and F. Somenzi. High-level power modeling, estimation, and optimization. In *Proc. 34th Design Automation Conf.*, pages 504–511, June 1997.

[21] D. Marculescu, R. Marculescu, and M. Pedram. Information theoretic measures for power analysis. In *IEEE Trans. on CAD*, pages 599–609, June 1996.

[22] R. Marculescu, D. Marculescu, and M. Pedram. Sequence compaction for power estimation: Theory and practice. In *IEEE Trans. on CAD*, pages 973–993, 1999.

[23] U. M. Maurer and J. L. Massey. Perfect local randomness in pseudo-random sequences. In *Advances in Cryptology - CRYPTO '89. Proceedings.*, pages 100–112, August 1989.

[24] S. Micali and C. P. Schnorr. Efficient, perfect random number generators. In *Advances in Cryptology - CRYPTO '88. Proceedings.*, pages 173–198, August 1988.

[25] A. Raghunathan, N. K. Jha, and S. Dey. *High-level power analysis and optimization*. Kluwer Academic Publishers, Norwell, MA, 1998.

[26] A. Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach*. Birkhauser, Boston, 1993.

[27] V. N. Yarmolik and S. N. Demidenko. *Generation and Application of Pseudorandom Sequences for Random Testing*. John Wiley and Sons, New York, 1988.