Randomness Enhancement Using Digitalized Modified Logistic Map

Shih-Liang Chen, *Student Member, IEEE*, TingTing Hwang, *Member, IEEE*, and Wen-Wei Lin

***Abstract*—In this brief, a nonlinear digitalized modified logistic map-based pseudorandom number generator (DMLM-PRNG) is proposed for randomness enhancement. Two techniques, i.e., constant parameter selection and output sequence scrambling, are employed to reduce the computation cost without sacrificing the complexity of the output sequence. Statistical test results show that with only one multiplication, DMLM-PRNG passes all cases in SP800-22. Moreover, it passes most of the cases in *Crush*, one of the test suites of TesuU01. When compared with solutions based on digitized pseudochaotic maps previously proposed in the literature, in terms of randomness quality, our system is as good as a Rényi-map-based PRNG and better than a logistic-map-based PRNG. Moreover, compared with solutions based on a Rényi-map-based PRNG, DMLM-PRNG is better scalable to high digital resolutions with reasonable area overhead.**

***Index Terms*—Discrete chaos, logistic map, pseudorandom number generator (PRNG).**

I. Introduction

PSEUDORANDOM number generators (PRNGs) are widely used in many applications, such as numerical analysis, integrated circuit testing, computer games, and cryptography. The quality of randomness is usually the main criteria to distinguish different PRNGs. Besides the quality of randomness, implementation cost and throughput are also important factors to evaluate the effectiveness of PRNGs in applications such as modern communication protocols.

Among them, chaotic map-based PRNGs have been proposed [1]–[3]. That the orbit of the chaotic system has the property of being irregular, aperiodic, and unpredictable and having sensitive dependence on initial conditions is useful in pseudorandom number generation. Moreover, in recent years, nonquantized chaotic maps have been used for generating true random numbers [4].

In [2], Li *et al.* proposed a logistic-map-based PRNG with timing-based reseeding method that replaced the last five least significant bits of the output sequence by a fixed pattern when a period of time is reached. Although Li's system [2] improves the randomness quality of the output sequence when compared with a classical logistic map, Li's system still has some statistical weak points in testing results of SP800-22 [5] and TestU01 [6].

In [3], Addabbo *et al.* proposed a low hardware complexity PRNG based on a piecewise-linear Rényi map. By utilizing the nonlinear property during truncation, Addabbo's system extended the period of Rényi map with length up to $2^n - 1$. The authors also provided a method to combine two subsystems to form a system that has maximum global cycle length and good quality of randomness. One disadvantage of the system is that it is not easy to scale the system to high precision. The first reason is that the maximum cycle length of the output sequence depends on the values of parameters that are not easy to find. Second, the quality of randomness is not predictable even when the system precision is increased. For example, in [3], a 24-bit system can pass statistical tests in SP800-22, but a 31-bit system, which is the largest precision reported in [3], fails 2 of 15 tests in the same test suite.

In this brief, we propose a PRNG that is based on a digitalized modified logistic map (DMLM). Similar to [7] and [8], DMLM is defined in $\gamma \geq 4$ to extend the parameter space. It is shown that the modified logistic map is a pseudochaotic map with larger parameter space as compared with a classical logistic map [7] and is suitable for secure communications [8]. However, high implementation cost renders it an unsuitable PRNG. In this brief, we will propose two techniques, i.e., constant parameter selection and output sequence scrambling, to reduce the computation cost without sacrificing the complexity of the output sequence. Moreover, we show that it is easy to extend the precision of DMLM-PRNG.

The statistical test results show that with only one multiplication, our system passes all cases in SP800-22 [5]. Moreover, it passes most of the cases in *Crush*, one of test suites of TestU01 [6], whereas Li's [2] and single Addabbo's [3] systems fail in almost all cases.

The rest of this brief is organized as follows. In Section II, DMLM-PRNG is presented. The parameter selection, scrambling method, and implementation issues will also be discussed. In Section III, we will compare DMLM-PRNG to other pseudochaotic map-based PRNGs with respect to statistical properties, implementation cost, and throughput. Finally, concluding remarks are given in Section IV.

II. Modified Logistic Map-Based PRNG

A. DMLM

A classical logistic map is defined by

$$x_{i+1} = \gamma x_i (1 - x_i), \quad x \in [0, 1]. \tag{1}$$

It is well known that a logistic map presents chaos for $3.57 < \gamma \leq 4$. However, the set of parameters γ's that forms short

Manuscript received April 13, 2010; revised July 6, 2010; accepted September 13, 2010. Date of current version December 15, 2010. This work was supported in part by the National Science Council under Grant NSC-97-2220-E-007-033 and Grant NSC-97-2220-E-007-043. This paper was recommended by Associate Editor Y. Horio.

S.-L. Chen and T.T. Hwang are with the Department of Computer Science, National Tsing Hua Unversity, Hsinchu 300, Taiwan (e-mail: chensl@cs.nthu.edu.tw; tingting@cs.nthu.edu.tw).

W.-W. Lin is with the Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: wwlin@math.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSII.2010.2083170

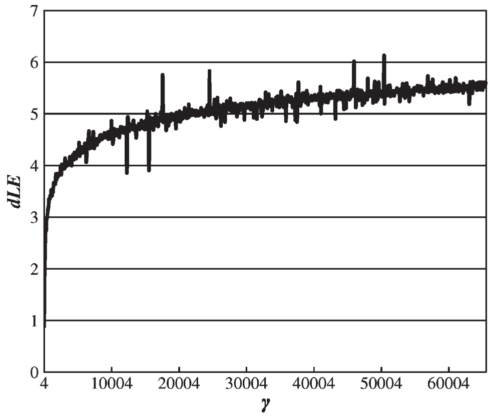

Fig. 1. *dLE*s of a 32-bit DMLM for $4 \leq \gamma \leq 2^{16}$.

periodic *windows* of (1) is open and dense. A periodic *window* is a set of parameters whose trajectories exhibit periodic behavior [9]. The trajectory generated by a parameter in a short periodic *window* easily enters a short cycle and is not suitable for applications in security. Moreover, the attractor is not fully distributed in [0, 1]. Thus, previous PRNGs based on logistic maps used γ close or equal to 4 for fast computation and high output complexity [2].

To use $\gamma \geq 4$ and digitalize a logistic map, a q-bit DMLM is defined by

$$x_{i+1} = \lfloor [\gamma x_i(1-x_i)] \,(\mathrm{mod}\ 1) \rfloor_q \tag{2}$$

where the operation $x(\mathrm{mod}\ 1)$ is used to keep x inside [0,1) by dropping the integer part of x, and $\lfloor x \rfloor_q$ is a truncation function to preserve the most significant q bits of x and drop others. Furthermore, a q-bit binary number x in [0, 1) can be presented by $x = \sum 2^{-j} x[j]$, where $x[j]$ is the jth bit of x. It holds that x_i in (2) is a q-bit binary number.

Next, to verify the pseudochaotic properties of DMLM, we conduct numerical experiments including *bifurcation analysis and discrete Lyapunov Exponent (dLE)* [10].

First, the bifurcation analysis is performed on 1000 γ's evenly selected in $4 \leq \gamma \leq 2^{16}$ for a 32-bit DMLM. In our experiment, we do not detect any short periodic *windows* in 1000 γ's.

Second, let $\alpha = \{x_0, x_1 = F(x_0), \ldots, x_{m-1} = F(x_{m-2})\}$ be a periodic orbit with period m, where F is defined in (2). Let $\delta = 2^{-q}$. The *dLE* [10] of the map F for a periodic orbit α is calculated by

$$\lambda_{(F,\alpha)} = \frac{1}{m} \sum_{k=0}^{m-1} \ln \frac{|F(x_k + \delta) - F(x_k)|}{\delta}. \tag{3}$$

Fig. 1 shows the *dLE*s of a 32-bit DMLM for 1000 γ's evenly selected in $4 \leq \gamma \leq 2^{16}$. For each γ, 1000 different initial conditions are given to generate 1000 periodic orbits. Then, the average *dLE* is reported. It shows that all *dLE*s are positive in $4 \leq \gamma \leq 2^{16}$. These numerical results indicate that DMLM presents pseudochaos with large parameter space when $\gamma \geq 4$.

To apply DMLM in PRNG, we will further define the value of γ. The objectives are twofold: 1) low-cost computation and 2) randomness quality. To reduce the computation cost, we focus on the subset of γ, where $\gamma = 2^k$. In this case, only one multiplication is needed to compute the orbit because the multiplication of $\gamma = 2^k$ only requires a shifting operation.

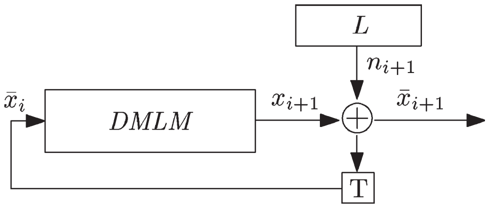

Fig. 2. Scrambling strategy for DMLM.

Next, to increase the randomness quality, we start from the point of implementing the multiplication in finite-precision arithmetic. Because the multiplication is defined in finite precision, truncation is needed after multiplying two numbers to store a product in the same number of bits. Let c be the $2q$-bit result of $x_i(1-x_i)$. Equation (2) can be rewritten as $x_{i+1} = c[k+1 : k+q]$ for $0 \leq k \leq q$. To have a uniformly distributed x_{i+1}, k should be close to $q/2$ so that the middle bits of c are preserved after truncation. (The reason will be explained in the next section.) Hence, our q-bit DMLM is reduced to

$$\mathrm{DMLM}(x_i) = \left\lfloor \left[2^{\lceil \frac{q}{2} \rceil} x_i(1-x_i) \right] (\mathrm{mod}\ 1) \right\rfloor_q . \tag{4}$$

The sequence generated by DMLM is defined by $x_{i+1} = \mathrm{DMLM}(x_i)$.

In hardware implementation, multiplying $2^{\lceil q/2 \rceil}$ can be implemented by a shift operation. Now, DMLM requires only one multiplication.

B. Scrambling Method

In this section, we discuss how to increase the cycle length. It is known that the length of an orbit generated by a digitalized pseudochaotic map is far below the total number of states. It is insufficient if a long random number sequence is required. To increase the cycle length, the method to scramble several least significant bits is useful and widely used. In [11], an external uniformly distributed PRNG is used as noise to scramble not only the output but also the parameter to increase the cycle length. In [12], a linear feedback shift register (LFSR), which has uniformly distributed outputs, is used to scramble the output of a digital chaotic system.

Similar to [12], we use LFSR to scramble the output of DMLM. Fig. 2 shows the proposed scrambling strategy, where L is a q-bit primitive LFSR, and T is a one-step delay block. The noise sequence generated by $n_{i+1} = L(n_i)$ is XORed with x_{i+1} to produce $\bar{x}_{i+1}$ before $\bar{x}_{i+1}$ is fed back to DMLM. The low bound of cycle length of the scrambled system is analyzed as follows. Let the scrambling period and the register length of an LFSR be Δ and l, respectively. In [12], the low bound of the cycle length of output sequence is specified by

$$\Delta \cdot (2^l - 1).$$

Generally, Δ should be small enough to prevent a trajectory from entering a short cycle loop. To increase the cycle length, we let $\Delta = 1$ and $l = q$. Hence, the low bound of cycle length will be $2^q - 1$. The overhead of the scrambling function is a q-bit LFSR.

Since the LFSR and the DMLM are not functionally independent, the transition behavior of the LFSR and the DMLM will affect the distribution of $\bar{x}_i$. The transition behavior of the scrambled system will be analyzed in the next section.

C. Property of the System

In this section, we will show that our constant parameter selection together with the scrambling method can produce uniformly distributed outputs, which is an important property of random number generators.

Our scrambled system can be defined as follows.

Definition 1: Let $\bar{x}_i$, x_i, and n_i be q-bit binary numbers. Let F and L be a pseudochaotic map and a primitive LFSR, respectively. Let $x_{i+1} = F(\bar{x}_i)$ and $n_{i+1} = L(n_i)$. The scrambled system is defined by $\bar{x}_{i+1} = D(\bar{x}_i, n_i) = F(\bar{x}_i) \oplus L(n_i)$.

Notice that in Definition 1, $F(\bar{x}_i)$ and $L(n_i)$ are functionally dependent and potentially statistically dependent. Although $L(n_i)$ is uniformly distributed, $F(\bar{x}_i) \oplus L(n_i)$ is not always uniformly distributed.

Assume that $\{\bar{x}_i[j]\}_{i>0}$ is selected as the binary output sequence. The objective is to prove that a uniformly distributed sequence $\{\bar{x}_i[j]\}_{i>0}$ is generated by the given system.

Let the state transition probability matrix of $D : \bar{x}_i[j] \rightarrow \bar{x}_{i+1}[j]$ be

$$T_D = \begin{bmatrix} d_{0,0} & d_{1,0} \\ d_{0,1} & d_{1,1} \end{bmatrix} \tag{5}$$

where $d_{\mu,\nu} = P(\bar{x}_{i+1}[j] = \nu | \bar{x}_i[j] = \mu)$ for μ, $\nu = 0, 1$. We define the fix-point condition for the discrete Frobenius–Perron equation for the transition D as

$$p = T_D \times p = [p_0\ p_1]^T \tag{6}$$

where $p_\mu = P(\bar{x}_i[j] = \mu)$ is the state probability for $\mu = 0, 1$.

For a uniformly distributed sequence $(\{\bar{x}_i[j]\}_{i>0})$, p in (6) should be equal to $[1/2\ 1/2]^T$.

The sufficient conditions for (6) with $p = [1/2\ \ 1/2]^T$ are discussed in the following properties.

Property 1: If $n_i[j]$ and $\bar{x}_i[j]$ are statistically independent, then (6) holds with $p = [1/2\ 1/2]^T$. In addition, T_D is a matrix with all entries equal to 1/2.

Proof: Let T_F and T_N be the state transition probability matrices of $F : \bar{x}_i[j] \rightarrow x_{i+1}[j]$ and $N : x_{i+1}[j] \rightarrow \bar{x}_{i+1}[j]$, respectively. T_F and T_N can be defined as follows:

$$T_F = \begin{bmatrix} f_{0,0} & f_{1,0} \\ f_{0,1} & f_{1,1} \end{bmatrix} \quad \text{and} \quad T_N = \begin{bmatrix} n_{0,0} & n_{1,0} \\ n_{0,1} & n_{1,1} \end{bmatrix} \tag{7}$$

where $f_{\mu,\nu} = P(x_{i+1}[j] = \nu | \bar{x}_i[j] = \mu)$ and $n_{\mu,\nu} = P(\bar{x}_{i+1}[j] = \nu | x_{i+1}[j] = \mu)$ for μ, $\nu = 0, 1$.

We can rewrite (6) by $p = (T_N \times T_F) \times p$. Because $L(n_i)$ is uniformly distributed, $n_{\mu,\nu} = 1/2$ for μ, $\nu = 0, 1$. Moreover, we know that $f_{0,0} + f_{0,1} = 1$ and $f_{1,0} + f_{1,1} = 1$. Thus, T_D is a matrix with all entries equal to 1/2. This implies $p = [1/2\ 1/2]^T$ is the stable probability of T_D. ■

Property 2: If $\bar{x}_i[j]$ and $x_i[j]$ are statistically independent, then $n_i[j]$ and $\bar{x}_i[j]$ are statistically independent. Therefore, the assertion of Property 1 holds.

Proof: From the assumption, we see that $F(\bar{x}_i)$ is a fully disturbing channel that can remove any statistical dependence between $\bar{x}_i[j]$ and $x_i[j]$. Furthermore, $n_i[j]$ and $x_i[j]$ are statistically independent. If $x_i[j]$ is uniformly distributed, then $n_i[j]$ and $\bar{x}_i[j]$ are statistically independent, which follows from the balanced property of the XOR operation. Thus, the assertion of Property 1 also holds. ■

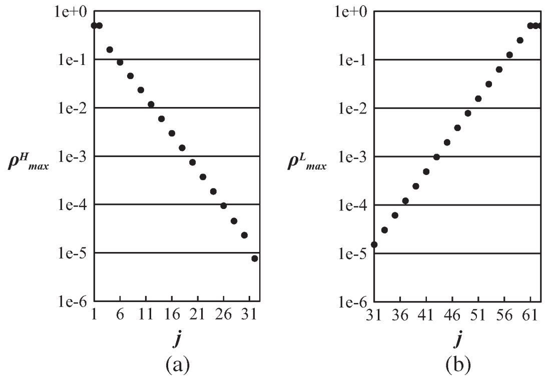

Fig. 3. (a)ρ^H_{max} for $1 \le j \le 32$. (b) $\rho^L_{\max}$ for $31 \le j \le 63$.

According to Properties 1 and 2, to have uniformly distributed outputs for the scrambled system, DMLM $(F(\bar{x}_i))$ should be a fully disturbing channel to remove any correlation between $\bar{x}_i[j]$ and $x_i[j]$. That is, the transition probability of T_F should be uniform.

In (2), γ is equal to 2^k. The objective of our constant parameter selection is to find a k so that the transition probability of (2) is close to uniform. Let $x'_i = (1 - x_i)$ and c be the $2q$-bit result of $x_i \times x'_i$. Equation (2) can be rewritten by $x_{i+1} = c[k+1 : k+q]$ for $0 \le k \le q$. Consider a case where $c[2q]$ is selected as the output sequence. Since $c[2q] = x_i[q] \times x'_i[q]$, $c[2q]$ is equal to 0 when $x_i[q]$ is 0. The high correlation between $c[2q]$ and $x_i[q]$ results in a nonuniform transition probability and makes $c[2q]$ an unsuitable output candidate. Similarly, high correlation exists between the most significant bits of $c(c[1])$ and $x_i\ (x_i[1])$.

On the contrary, the middle bits of c computed by more number of partial products (as compared with the most/least significant bits of c) and carry-in bits depend on each bit of x_i more uniformly.

An experiment is conducted to understand correlations between bits of c and x_i with $q = 32$. The state transition probability matrix going from the tth bit of $x_i\ (x_i[t])$ to the jth bit of $c\ (c[j])$ can be defined by a 2-by-2 matrix T_F, where the entry at the μth column and νth row shows the probability $f_{\mu,\nu} = P(c[j] = \nu | x_i[t] = \mu)$, where μ, $\nu = 0, 1$.

The following equation is used to measure the maximum distance between $f_{\mu,\nu}$ and 1/2 for each T_F for different values of t and j:

$$\rho_{\max} = \max\left(\left| f_{\mu,\nu} - \frac{1}{2} \right|\right) \tag{8}$$

where μ, $\nu = 0, 1$.

The smaller $\rho_{\max}$ is, the more uniform transition probability T_F has. Two sets of $\rho_{\max}$ are calculated. The first set is $\rho^H_{\max}$, which is the transition probability from the most significant bit $x_i[1]$ to $c[j]$ for $1 \le j \le 32$. The second set is $\rho^L_{\max}$, which is the transition probability from the least significant bit $x_i[32]$ to $c[j]$ for $31 \le j \le 63$. As shown in Fig. 3, $\rho^H_{\max}$ and $\rho^L_{\max}$ are decreasing when j is close to 32. It shows that $f_{\mu,\nu}$ is close to 1/2 when the middle bits of c are selected. Please notice that $\rho_{\max}$'s smaller than 10^{-6} are not shown in the graph.

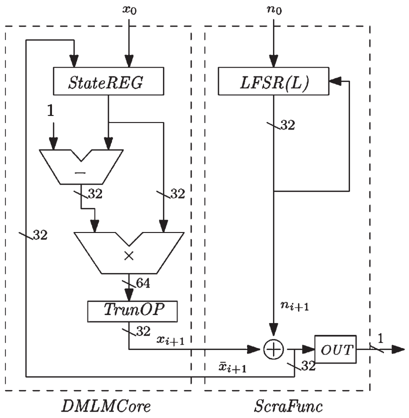

Fig. 4. Architecture of DMLM-PRNG.

Note that the above discussion is still empirical because the result shown in Fig. 3 is not a rigid statistical analysis. However, the discussion could provide some insights into this special case of pseudorandom number generation.

Hence, to have uniform transition probabilities from x_i to x_{i+1} in (2), k should be close to $q/2$ to preserve the middle bits of the product after truncation. We know that our constant parameter selection for DMLM will produce more uniformly transition probabilities. With our constant parameter selection and the scrambling method, DMLM-PRNG can generate uniformly distributed outputs.

D. Implementation

We take a 32-bit DMLM-PRNG as an example to show the efficiency of our DMLM-PRNG implementation. The structure of the DMLM-PRNG is shown in Fig. 4. The first module *DMLMCore* generates the state value of the DMLM. The second module *ScraFunc* scrambles the state value by a noise generated by $LFSR(L)$. The scrambled state value will be fed back to *DMLMCore*. Finally, *OUT* is the output generator that selects the most significant bit of $\bar{x}_{i+1}$ to be the random number sequence.

The current state value of DMLM-PRNG is stored in a 32-bit register *StateREG*. A 32-bit subtractor is used to compute $(1 - x_i)$. To compute the next state, only one multiplier is needed for $x_i(1 - x_i)$. Operations to multiply 2^{16} and truncate the result of $x_i(1 - x_i)$ to 32 bits are implemented by signal selection in *TrunOP*. Since the truncation operation drops the most significant 16 bits during multiplication (i.e., the logic circuit for these bits is no longer needed and can be removed), the required area is only a 24 × 32 multiplier as compared with the area of a 32 × 32 multiplier needed by a classical logistic map. The detailed performance evaluation of DMLM-PRNG will be discussed in the next section.

III. Performance Evaluation

In this section, we will compare DMLM-PRNG to other pseudochaotic map-based PRNGs with respect to statistical properties, implementation cost, and throughput.

To evaluate the randomness of our system, two test suites, i.e., NIST SP800-22 [5] and TestU01 [6], are used. The SP800-22 test suite has been the standard reference for PRNGs.

TABLE I
Testing Results of DMLM-PRNG in Different Precisions by SP800-22

Tests	20-bit		24-bit		32-bit	
	Yield	U-value	Yield	U-value	Yield	U-value
Frequency	0.9917	0	0.9917	0.15520	0.9917	0.00619
Block freq.	0.9917	0	0.9750	0.02381	0.9750	0.33716
Cumulative*	0.9917	0	0.9791	0.31171	0.9917	0.46264
Runs	0.9833	0	0.9917	0.28730	0.9917	0.12837
Longest run	0.9833	0.09561	0.9833	0.50093	0.9917	0.88813
Rank	0.9833	0.00699	0.9917	0.42203	0.9833	0.46859
FFT	0.9750	0.25355	0.9667	0.50093	0.9750	0.16260
Nonoverlap.*	0.9791	0.23426	0.9805	0.48152	0.9816	0.49706
Overlap.	0.9833	0.00095	0.9667	0.98088	0.9917	0.80433
Universal	0.9667	0.01341	0.9833	0.23276	0.9833	0.32418
Apen	0.9833	0.36414	0.9833	0.00887	0.9833	0.48464
Random e.*	0.9928	0.41009	0.9834	0.29321	0.9945	0.14739
Random e.v.*	0.9968	0.08301	0.9877	0.00788	0.9983	0.00131
Serial*	0.9875	0.63531	0.9833	0.39006	0.9833	0.28810
Linear Comp.	0.9917	0.60245	0.9833	0.78872	0.9667	0.00836
Failure Counts	0	4	0	0	0	0

*average result of multiple tests is shown.

We use SP800-22 as our first randomness test. Then, TestU01 is used for further comparing randomness between all systems.

The configuration for the SP800-22 test is as follows. For each test, 120 sequences will be generated by DMLM-PRNG with the length of 10^6 bits. For each sequence, each test will produce a P-*value* from SP800-22. The P-*value* should be in the range [0.01, 1.00] to pass the test. As suggested in SP800-22, for each test, the minimum pass rate of a pseudorandom source is 0.9627 out of 120 sequences. The U-*value* is also reported for the distribution measurement of the collected P-*values*. If the U-*value* is greater than 10^{-4}, then the sequences can be considered to be a pseudorandom sequence with acceptable quality of randomness. For TestU01 test, as recommended by [6], *SmallCrush* including 15 subtests is taken as a fast check of the basic randomness requirement. Next, *Crush* needs sequences with 2^{35} bits to perform 144 tests. For each test, a *P-value* is calculated and should be in [0.001,0.999] to pass the test.

We first conduct an experiment to understand the quality of randomness of DMLM-PRNG when the system precision is increased. As shown in Table I, systems in 20, 24, and 32 bits are tested. When the system precision is 20 bits, the proposed PRNG passes all tests but has four failure U-values. However, when the system precision is larger than 24 bits, DMLM-PRNG passes all tests in SP800-22 and has uniformly distributed P-*values* for each test. The experimental results show that the statistical properties of DMLM-PRNG are becoming better when we increase the system precision.

The discussion in Section II-C has shown that the middle bits of $\bar{x}_i$ are those with the best randomness properties. We compare the randomness properties of two cases, where the first bit and the 16th bit of $\bar{x}_i$ are respectively selected as output sequences. When the precision is 20 bits, the failure count of the U-value is indeed reduced from 4 to 2 for the middle bit case.

1) Randomness Improvement By Scrambling Function: To understand the efficiency of the scrambling function in DMLM, tests with/without scrambling function are performed. The failure counts of tests in different testing suites are shown in Table II. In row labeled DMLM, without the scrambling function, DMLM fails some tests because of the short output length. On the contrary, in row labeled DMLM-PRNG, with the scrambling function, the 32-bit DMLM-PRNG passes all tests

TABLE II
FAILURE COUNTS IN STATISTICAL TESTS FOR DIFFERENT SYSTEMS

System	Precision	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)
DMLM (W/O scrambling)	32	9	14	139
DMLM-PRNG	32	0	0	3
Classical Logistic Map	32	9	15	140
Addabbo's [3]	31	2	14	122
Li's [2]	32	1	15	144
Addabbo's [3](combined)	32	0	0	3

TABLE III
FAILURE COUNTS IN STATISTICAL TESTS WITH SCRAMBLING FUNCTION

System	Number of Registers	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)
DMLM-PRNG	32+32	0	0	3
Classical Logistic Map	32+32	8	14	124
Li's [2]	42+32	12	13	123
Addabbo's [3]	31+31	0	1	14
Classical Logistic Map (no-scr.)	64	0	0	27

in SP800-22 and *SmallCrush* and almost all tests in *Crush* test suites.

2) Comparison With Previous Work: In addition, in Table II, test results for the 32-bit DMLM-PRNG are compared with those for *Classical Logistic Map*, Addabbo's [3] system, and *Li's* [2] system. For Addabbo's system, the precision for the experiment is 31 bits because it is the largest precision reported in [3]. In the last row, the results for the combined Addabbo's system of 17- and 15-bit subsystems are also included. This table shows that both DMLM-PRNG and combined Addabbo's systems have good statistical properties.

3) Adding Scrambling Function to Other Systems: The total numbers of states in each testing system are different. To provide each system comparable number of states, the scrambling function is applied to other systems in the same way that it is applied to DMLM-PRNG. In Table III, testing results show that *Classical Logistic Map* and *Li's* systems still fail lots of tests even when the number of registers is doubled. Moreover, *Li's* system is worse than its nonscrambling version. Similar to DMLM-PRNG, Addabbo*'s* system can improve the quality of randomness by scrambling. The table also shows that DMLM-PRNG performs slightly better than Addabbo*'s* system in terms of failure count. Moreover, results for a nonscrambled 64-bit classical logistic map are also reported in the row labeled *Classical Logistic Map (no-scr.)*. It shows that the quality of randomness can be improved by precision increase (see a 32-bit version in Table II), but the trajectory of a digitalized logistic map eventually enters a loop with unknown length. The quality of randomness of a 64-bit classical logistic map is worse than that of DMLM-PRNG in terms of failure count.

The transition probabilities of Addabbo's system are close to uniform, whereas those of the classical logistic map and Li's systems are not. Hence, when the scrambling function is used, it results in good result for Addabbo's system but worse statistic properties in the classical logistic map and Li's systems.

TABLE IV
COMPARISONS OF DATA PATH COMPONENTS, AREA, AND THROUGHPUT

PRNGs	Multipliers	Counters	LFSRs	Area (2-NAND)	ThroughPut (bits/sec)
DMLM-PRNG	1(24x32*)	0	1(32-bit)	9622	200M
Li [2]	1(32x32)	1(10-bit)	0	20075	200M
Addabbo [3]	1(31x31)	0	0	N.A.	N.A.

*multiplier with the same area cost.

Finally, we compare cost of data-path in different systems. DMLM-PRNG is implemented and synthesized with TSMC 18 μm technology library. As shown in Table IV, in a 32-bit DMLM-PRNG, one 24 × 32 multiplier and a 32-bit LFSR are required. From [2], the gate count for *Li's* system is calculated by total gate area divided by a two-input NAND gate that is equal to 9.98 μm^2. The comparison of area cost in terms of gate counts for DMLM-PRNG and other systems is shown in the column denoted by *Area*. The area cost of DMLM-PRNG is 47.9% of that of *Li's* system. Compared with *Li's* system, DMLM-PRNG has smaller area and more complex output sequence with the same throughput. As compared with Addabbo*'s* system, DMLM-PRNG is easy to scale to large precision with reasonable area overhead.

IV. CONCLUSION

In this brief, we have proposed a nonlinear DMLM-PRNG. With our constant parameter selection and scrambling method, DMLM-PRNG has output sequence with good randomness quality at low implementation cost. Statistical test results have shown that the randomness quality of DMLM-PRNG is as good as Addabbo's [3] combined system and better than Li's [2] system. Moreover, our system has shown better scalability than Addabbo's system.

REFERENCES

[1] X. Wang, J. Zhang, and W. Zhang, "Chaotic keystream generator using coupled NDFs with parameter perturbing," in *Proc. Int. Conf. Cryptology Netw. Security*, 2006, vol. 4301, pp. 270–285.

[2] C. Y. Li, J. S. Chen, and T. Y. Chang, "A chaos-based pseudo random number generator using timing-based reseeding method," in *Proc. IEEE ISCAS*, 2006, pp. 21–24.

[3] T. Addabbo, M. Alioto, A. Fort, A. Pasini, S. Rocchi, and V. Vignoli, "A class of maximum-period nonlinear congruential generators derived from the Rényi chaotic map," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 4, pp. 816–828, Apr. 2007.

[4] S. Callegari, R. Rovatti, and G. Setti, "Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos," *IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 793–805, Feb. 2005.

[5] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST, Gaithersburg, MD, Tech. Rep. 800-22, 2001.

[6] P. L'Ecuyer and R. Simard, "Testu01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 1–22, Aug. 2007.

[7] S.-M. Chang, M.-C. Li, and W.-W. Lin, "Asymptotic synchronization of robust hyper-chaotic systems and its applications," *Nonlinear Anal. Real World Appl.*, vol. 10, no. 2, pp. 869–880, Apr. 2009.

[8] S.-L. Chen, S.-M. Chang, T.T. Hwang, and W.-W. Lin, "Digital secure-communication using robust hyper-chaotic systems," *Int. J. Bifurc. Chaos*, vol. 18, no. 11, pp. 3325–3339, Nov. 2008.

[9] S. H. Strogatz, *Nonlinear Dynamics and Chaos*. Boulder, CO: Westview, 1994, p. 356.

[10] L. Kocarev, J. Szczepanski, J. M. Amigo, and I. Tomovski, "Discrete chaos—I: Theory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 6, pp. 1300–1309, Jun. 2006.

[11] J. Cermák, "Digital generators of chaos," *Phys. Lett. A*, vol. 214, no. 3/4, pp. 151–160, May 1996.

[12] T. Sang, R. Wang, and Y. Yan, "Perturbance-based algorithm to expand cycle length of chaotic key stream," *Electron. Lett.*, vol. 34, no. 9, pp. 873–874, Apr. 1998.