電気学会論文誌 D（産業応用部門誌）
IEEJ Transactions on Industry Applications
Vol.145 No.4 pp.192–199　DOI: 10.1541/ieejias.145.192

論　文

画像データと LFSR による乱数生成手法

上級会員　市川　周一*a)

Random Number Generation from Image Data and LFSR

Shuichi Ichikawa*a), Senior Member

（2024年2月6日受付，2024年4月8日再受付）

In 2021, Masaoka, Ichikawa, and Fujieda proposed generating unpredictable random numbers by sampling a hardware LFSR (Linear Feedback Shift Register) at uneven intervals, where the source of entropy was the fluctuation of sampling intervals. Then, in 2023, Chiba and Ichikawa generated random numbers that passed the NIST test by utilizing weather data downloaded from the Internet as the source of entropy. This study proposes the use of digital images as the entropy source for random number generation. Our evaluation results indicate that a 64-bit LFSR should be used, and that fluctuations of the sampling intervals should be generated with the least significant bit of each pixel value. The random numbers generated by the proposed method passed both the Diehard and NIST SP 800-22 tests.

キーワード：乱数，URNG (Unpredictable Random Number Generator), LFSR (Linear Feedback Shift Register)
Keywords: random number, URNG (Unpredictable Random Number Generator), LFSR (Linear Feedback Shift Register)

1. はじめに

乱数生成は，セキュリティやシミュレーションなど多くの分野で必須の技術である。物理現象から生成される真性乱数（TRN；True Random Number）は予測不能であるが，生成するために専用のハードウェア（TRNG；True Random Number Generator）が必要になる。一方，確定的アルゴリズムにより数値的に生成される疑似乱数（PRN；Pseudo Random Number）は，ソフトウェアだけで生成できるが，初期状態とアルゴリズムから将来値を予測することができる。

Suciu ら[1][2] は，TRN と PRN の中間的な乱数として URN（Unpredictable Random Number）を提案した。Suciu らの URNG（Unpredictable Random Number Generator）は，パフォーマンスカウンタ（PFC；Performance Counter）を利用して，プロセッサの内部状態から乱数を生成する。PFC の値は読み出すタイミングや実行環境に依存するため，URNG の出力が実質的に予測不可能になる。ただし PFC の生成するエントロピーは小さいため，Suciu らの URNG で高品質の乱数を生成するには，複数 PFC の同時利用，複数プロセスの同時実行，後処理等の工夫が必要になる。

正岡・市川・藤枝[3] は，エントロピーの小さい PFC の代わりに，疑似乱数生成器の一種である LFSR（Linear Feedback Shift Register）を使用した。ソフトプロセッサに 128 bit の LFSR を追加し，適切な条件下で LFSR の下位 32 bit を読み出すことにより URN を生成した。この URNG のエントロピー源は，LFSR を読み出す間隔の揺らぎである。正岡らは実システムで URNG を実装評価し，128-bit LFSR で生成した URN が乱数検定に合格することを示した。

正岡ら[3] は，URNG に必要な LFSR のビット数や帰還多項式について，詳細な検討を行わなかった。そこで鴨狩と市川[4] は，LFSR を用いた URNG の構成方法について，シミュレーションで検討した。LFSR の仕様とサンプリング間隔を変化させて乱数品質を調査し，URNG の設計要件と使用条件を示した。

正岡ら[3] の URNG は，組込みプロセッサ内の LFSR（ハードウェア）を前提としているため，LFSR を持たない通常のプロセッサでは実現することができない。しかし鴨狩と市川[4] のシミュレーションで分かる通り，適切なエントロピー源があればソフトウェアの LFSR でも高品質な乱数列を生成することができる。

a) Correspondence to: Shuichi Ichikawa. E-mail: ichikawa@ieee.org
* 豊橋技術科学大学 電気・電子情報工学系
〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘 1-1
Department Electrical and Electronic Information Engineering, Toyohashi University of Technology
1-1, Hibarigaoka, Tempaku-cho, Toyohashi, Aichi 441-8580, Japan

現在，多くのシステムには通信機能が実装されており，ネットワークへの接続が可能になっている。ネットワーク上のデータからエントロピーを得ることができれば，ソフトウェアだけで URNG が実現できるはずである。例えば Fernandez ら[5] は，ツイッターのストリームから乱数列を生成する手法を提案している。

千葉と市川[6] は，ネットワーク上で提供されている気象観測データをエントロピー源として，LFSR を用いた URNG を実現することを提案した。気象は自然現象であり，気象データは一定量のエントロピーを含むことが期待される。千葉と市川は，中部地方 25 か所の風向データ 10 年分をエントロピー源として用い，生成された乱数列が NIST テスト[7] に合格することを示した。さらに別役と市川[8] は，乱数生成に必要な気象データ量が削減可能であることを示した。

本研究では，以上の先行研究の結果を踏まえて，新たに画像情報をエントロピー源とした URNG について検討する。近年ではカメラモジュールの小型化・高性能化・価格低下が著しく，多くのデバイスにカメラ機能が搭載されている。スマートホン・タブレット端末・ラップトップ PC はもちろん，自動車・建物・家電製品などにもカメラが組み込まれている。カメラの高解像度化により画像 1 枚の情報量が増大し，動画撮影が普及して画像の生成速度も向上しているため，エントロピー源としての有用性は大きいと期待される。また各機器は様々な環境で用いられるため，生成画像の多様性が大きく，それにより生成される URN の予測が困難になると考えられる。

なお本論文は，著者による研究会発表[9] を基に，NIST テスト[7] の評価結果などを加筆したものである。

2. LFSR を用いた URNG

本研究では，先行研究[3][4] に従い，LFSR を用いて URN 生成を行う。具体的には，LFSR のサンプリング間隔を画素データで変動させることにより，URN 系列を生成する。画素データは 1 次元配列に格納するものとし，以下の説明で配列名を *data*，配列サイズを N と表現する。画像サイズが 256×256 であれば，$N = 256 \times 256 = 65536$ となる。

8-bit LFSR の例を Fig. 1 に示す。LFSR は帰還多項式に対応するタップシーケンスで表現され，Fig. 1 のタップシーケンスは [8, 6, 5, 4] である。鴨狩と市川[4] の研究結果を踏まえて，本研究では 32-bit LFSR（タップシーケンス [32, 25, 15, 7][11]）と 64-bit LFSR（タップシーケンス [64, 45, 31, 14][11]）を用い，値の下位 32 bit をサンプルして URN として用いる。

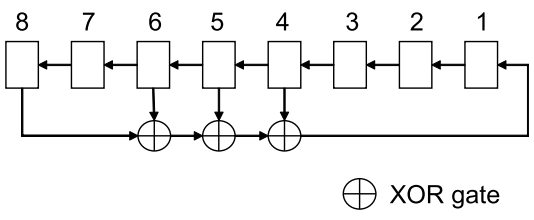

Fig. 1. An example of 8-bit LFSR [8, 6, 5, 4][4].

$32 < k < 64$ の任意の k-bit LFSR を使用することも可能だが，ソフトウェアでの実装を考慮した場合，レジスタや変数の長さは通常 2 冪 bit である。ハードウェア実装においても，LFSR の論理規模は小さいので，k を最小化する必要性は小さいと考えられる。そこで本研究では，32-bit と 64-bit LFSR の 2 種類について検討を行う。

LFSR の値を読む際に，n 回目のサンプリング間隔 $S(n)$ を以下の式で決める。サンプリング間隔とは，$(n-1)$ 回目のサンプルのあと帰還多項式を $S(n)$ 回適用して，n 回目のサンプルを行うという意味である。

$$S(n) = \alpha(n) + \beta \quad \cdots\cdots (1)$$

$$\alpha(n) = data[n \mod N] \mod 2^m \quad \cdots\cdots (2)$$

ここで定数 β は基本となるサンプリング間隔，$\alpha(n)$ はサンプリング間隔の揺らぎである。画素数よりサンプリング回数が大きい場合（$n \geq N$）もあるので，使用する画素値は $data[n \mod N]$ で求め，その値の下位 m bit を揺らぎ $\alpha(n)$ として用いる。従って $0 \leq \alpha(n) \leq 2^m - 1$ となる。

$m = 0$ は $\alpha(n) = 0$，即ち $S(n) = \beta$（定数）を意味している。m が大きいほどサンプリング間隔の揺らぎが大きくなることが期待されるが，一般に画素値の上位ビットは変化に乏しく，揺らぎの効果が小さいため乱数品質への寄与も小さい。一方で m が大きいと $S(n)$ が増大して計算時間が増える。これらの理由から，本研究では $0 \leq m \leq 4$ の範囲で評価を行う。

3. 画像データの使用法

画像処理の研究では，評価用に SIDBA（Standard Image Database）の画像を用いることが多い。そこで本研究でも，SIDBA の標準画像を評価に用いる。以下の評価で用いるサンプル画像を，Fig. 2 にまとめる。

SIDBA には様々なサイズの画像があるが，本研究では画素数による乱数品質の差を比較するため，(a) 256×256, (b) 512×512, (c) 1024×1024 のモノクロ画像を評価した。被写体によりエントロピーに差があることが予想されるが，256×256 では人物画像，512×512 では風景画像（漁船），1024×1024 は航空写真（空港）を選択している。Fig. 2(d) は，512×512 のモノクロ風景画像であるが，これはエントロピーの小さい画像の例を示すために取り上げ

(a) Lenna (256×256) (b) Fishing Boat (512 × 512) (c) Airport (1024×1024) (d) Steam and Bridge (512 × 512)

Fig. 2. Four sample images from SIDBA (Standard Image Database).

た（〈4·2〉節参照）。

乱数検定で必要な乱数列は大きいため，LFSR のサンプリング回数は画素数より大きくなる。画像ファイルから読みだした画素値は 1 次元配列に格納し，格納順で循環的に使用している。

SIDBA にはカラー画像も用意されているが，本研究の手法ではモノクロとカラーの本質的な差はない。カラー画像は 1 画素につき RGB の 3 値をもつため，データ量が同サイズのモノクロ画像の 3 倍になるだけである。

4. Diehard テストによる評価

乱数品質を評価するには一般に大きなデータ量が必要になる。例えば Diehard テスト[10] では，約 10^8 bit のデータ量が必要である。また NIST テスト[7] では，1 回のテストで約 10^6 bit を使用し，そのテストを 1000 回以上行うことが推奨されているので，合計 10^9 bit 程度が必要になる。

本章では，先行研究[3][4] と同様に Diehard テスト[10] を利用して乱数品質の評価を行う。Diehard テストは NIST テスト[7] ほど厳格ではないが，必要なデータ量が少ないため試行錯誤の多い研究段階に適している。

〈4·1〉 評価方法 Diehard テストは全 18 種のテストからなり，各テストで 1〜100 個（合計 313 個）の p 値を出力する。合格判定の基準は定められておらず，利用者の判断に委ねられているので，本研究でも先行研究[3] の基準を参考にして，以下の方法で乱数品質を評価する。

入力が理想的乱数であれば p 値は区間 $[0, 1)$ で一様に分布することが期待されるので，Table 1 に示した基準で各 p 値の成功（PASS）／弱成功（WEAK）／失敗（FAIL）を判定する。FAIL の発生確率（期待値）は 2×10^{-6} なので，入力が乱数であれば（ほぼ）発生しない。WEAK の発生確率（期待値）は 1×10^{-2} なので，WEAK が 1% 程度発生することは正常である。

単純に 313 個の p 値について PASS/WEAK/FAIL の個数を示すと，多くの p 値を出力するテストのウエイトが大きく見えてしまう。そこで以下の方法により，各テストの結果を判定する。各テストで出力される p 値の個数が 3 個以上であれば，得られた p 値の分布が一様であるかどうかの判定を Kolmogorov-Smirnov 検定（KS 検定）により行い，得られた p 値を Table 1 に示した基準で判定する。テストの出力する p 値が 2 個以下であれば，以下に述べる方法で結果を判定する。各テストで出力される p 値に，ひとつでも FAIL が含まれれば，そのテストは FAIL。出力される p 値に FAIL はなく全て WEAK であれば，そのテストは WEAK。それ以外の場合，そのテストは PASS とする。

こうして計算した全 18 テストの結果（PASS/WEAK/FAIL の内訳）により，乱数列の品質を評価する。

Fig. 3 と Fig. 4 は，2 章で説明した 32-bit LFSR と 64-bit LFSR を Diehard テストで評価した結果である。横軸は基本サンプリング間隔 β であり，揺らぎ $\alpha(n) = 0$ として乱数列を生成している。縦軸はテスト数で，内訳を PASS/WEAK/FAIL で示している。

Table 1. Diehard evaluation criteria[3].

Decision	Condition
PASS	$0.005 \le p < 0.995$
WEAK	$0.000001 \le p < 0.005$, or $0.995 \le p < 0.999999$
FAIL	$p < 0.000001$, or $0.999999 \le p$

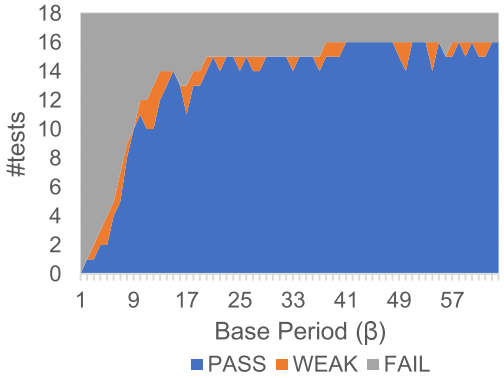

Fig. 3. Diehard test results of the 32-bit LFSR.

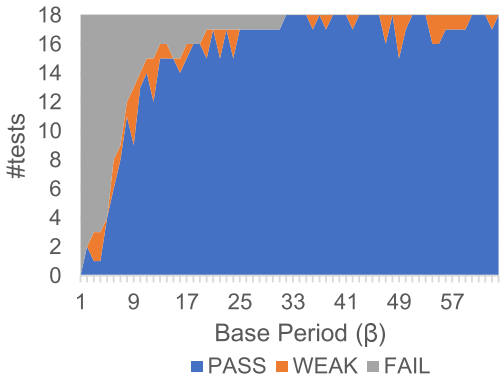

Fig. 4. Diehard test results of the 64-bit LFSR.

いずれも基本間隔 β が増加すると乱数品質が向上するが，32-bit LFSR（Fig. 3）では全てのテストに合格することができない。32-bit LFSR の $38 \le \beta \le 64$ で FAIL するテストは，Binary Rank（31×31）と Binary Rank（32×32）の 2 つ†である。32-bit LFSR の周期は $2^{32} - 1$ であるが，その周期では Binary Rank テストに合格できない。64-bit LFSR（Fig. 4）では，β の増大とともに乱数品質が向上し，$\beta \ge 32$ で 18 種全てのテストに合格している。これは，鴨狩と市川[4] の報告に整合する結果である。

本研究では，画像，パラメータ m，LFSR 長 (32, 64) を変えて評価を行うため，Fig. 3，Fig. 4 のように PASS/WEAK/FAIL の内訳全てを示すと記述が冗長になる。以下では最も重要な「FAIL の数」に着目して説明を行う。

〈4·2〉 32-bit LFSR の場合 Fig. 5〜Fig. 7 は，32-bit LFSR に対して，サイズの異なる 3 画像（Lenna，Fishing Boat，Airport）で揺らぎ $\alpha(n)$ を加えて乱数を生成し，得られた乱数列を Diehard テストで評価した結果をまとめたも

† $\beta = 56$ では Binary Rank に加えて Overlapping Sums テストが FAIL し，合計 FAIL 数が 3 となっている。

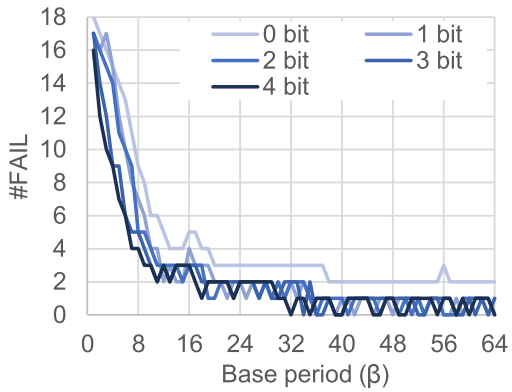

Fig. 5. Diehard test results of the Lenna (32-bit LFSR).

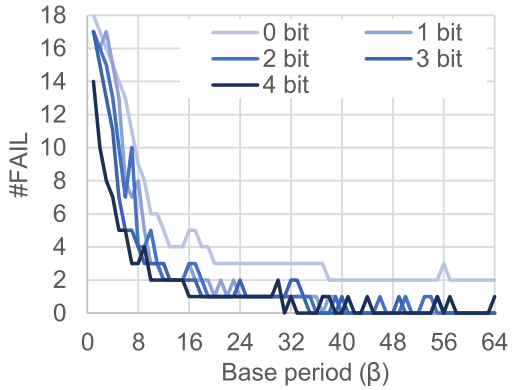

Fig. 6. Diehard test results of the Fishing Boat (32-bit LFSR).

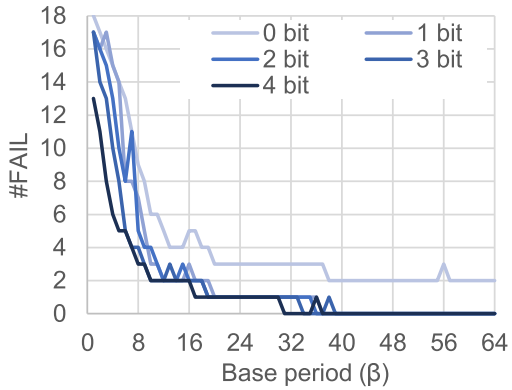

Fig. 7. Diehard test results of the Airport (32-bit LFSR).

のである。縦軸は FAIL の数，横軸は基本サンプリング間隔 β である。使用した下位ビット数を m bit（$0 \le m \le 4$）と表示して，それぞれの結果を示している。

Fig. 3 でも示した通り，揺らぎなし（$m = 0$）では β を大きくしても FAIL が消えない。揺らぎあり（$1 \le m \le 4$）では乱数品質が向上しているが，画像サイズが小さい場合（256×256）では乱数品質が不十分である。画像サイズが大きくなると乱数品質が向上し，1024×1024 では $\beta \ge 39$ で FAIL が 0 になっている。このことから，画像を用いてサンプリング間隔の揺らぎを与えると，乱数品質が向上することが確認できた。

一方，$1 \le m \le 4$ において，乱数品質の大きな差は見られなかった。同じ β において，m が大きい方が FAIL 数が小さいように見えるが，それは $S(n)$ の平均値の差によるものである。$m > 0$ では $\alpha(n) \ge 0$ となるので，同じ β でもサンプリング間隔 $S(n)$ は異なる。$\alpha(n)$ の平均値は画像データによって異なるが，Fig. 5〜Fig. 7 における $\alpha(n)$ の平均値は Table 2 に示す通りである。

Table 2. Average of the $\alpha(n)$ for $1 \le m \le 4$.

Average (bit)	$m = 1$	$m = 2$	$m = 3$	$m = 4$
Lenna	0.50	1.50	3.48	7.52
Fishing Boat	0.52	1.49	3.57	7.82
Airport	0.51	1.52	3.53	7.50
Stream and Bridge	0.62	1.35	3.34	7.21

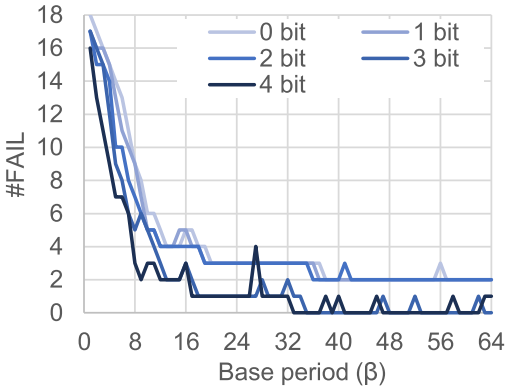

Fig. 8. Diehard test results of the Stream and Bridge (32-bit LFSR).

例えば Fig. 5 において，(β, m) における FAIL 数を $F(\beta, m)$ と表すなら，$F(8,0) = 8$，$F(8,1) = 7$，$F(8,2) = 5$，$F(8,3) = 5$，$F(8,4) = 4$ であるため，m が大きい方が良いように思われる。しかし $S(n)$ の平均値を揃えて比較するのであれば，比較すべきなのは $F(16,0) = 5$，$F(15,1) = 2$，$F(14,2) = 3$，$F(12,3) = 3$，$F(8,4) = 4$ であるため，$m = 1$ が最良の選択となる。

さらに，画像によっては乱数品質の向上が得られない場合がある。簡単な例は，全画素の値が 0 である「均一な黒」画像であろう。この場合は常に $\alpha(n) = 0$ であるため，乱数品質の向上は得られない。「均一な黒」は極めて特殊な画像ではあるが，SIDBA の標準画像にも問題のある画像が含まれている。

Fig. 8 は，画像「Stream and Bridge」（Fig. 2(d)）による評価結果である。この画像は 512×512 であるが，同サイズの「Fishing Boat」（Fig. 6）と比べて FAIL が多い。特に $m = 1$〜2 における FAIL 数が $m = 0$ とほぼ同じで，品質改善が見られない。この原因は，本画像の画素値が特殊な分布をしているためである。

画像「Stream and Bridge」のヒストグラム（Fig. 9）には，明らかに不自然なトーンジャンプが認められる。画素値（8 bit）の取り得る 256 階調のうち，画像に出現する画素値は 64 通りであり，残る 192 通りの出現度数は 0 となっている。Fig. 9 に描かれた 64 本のバーは，画像に出現

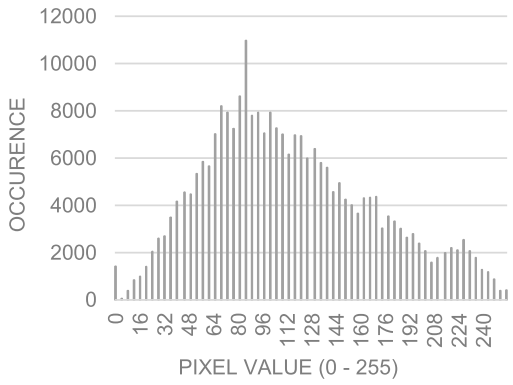

Fig. 9. Histgram of the Stream and Bridge.

する画素値（64 通り）に対応している。画像に出現しない画素値（192 通り）はバーの間の空白として描かれているが，バーの間隔は一定ではない。間隔は概ね 4 であるが，所々で 5 になっている。そのため画素値の下 2 ビットの出現頻度を調べると，0 が 7.7%，1 が 56.0%，2 が 29.9%，3 が 6.3% であり，著しい偏りが発生している。このような画像では，提案手法による乱数品質改善が制約を受けると思われる。他の 3 画像（Lenna，Fishing Boat，Airport）の階調は連続的で，Stream and Bridge のような著しい偏りは見られない。

〈4·3〉 64-bit LFSR の場合 〈4·2〉節で見てきたように，32-bit LFSR では，サンプリング間隔の揺らぎなしでは乱数検定に合格できない。従って「均一な黒」のような問題画像を与えられた場合は，乱数品質が低下して応用に問題が生じる可能性がある。

こうした問題を回避するには，揺らぎなしでも乱数検定に通過する乱数列を生成すればよい。具体的には，64-bit LFSR を使用すれば，$\beta \geq 32$ で全てのテストに合格することができる（Fig. 4）。揺らぎなしで合格できる構成であれば，「均一な黒」のような画像でも乱数列は検定に合格するため，安心して使用することができる。

32-bit LFSR では揺らぎにより乱数品質が向上したため，64-bit LFSR でも $m \geq 1$ では $m = 0$ より乱数品質が高いと期待できる。しかし実験による裏付けが必要なので，64-bit LFSR でも 32-bit LFSR と同様の実験を行った。

Fig. 10〜Fig. 12 は，64-bit LFSR で提案手法を評価した結果である。Lenna では $\beta \geq 39$，Fishing Boat では $\beta \geq 32$，Airport では $\beta \geq 32$ で，全てのテストに合格している。

Airport（Fig. 12）の $(\beta, m) = (61, 2)$ で FAIL が発生しているが，これは OPERM5 の FAIL であった。Diehard テストの OPERM5 にはバグがあることが知られており[12]，先行研究[4] においても不可解な FAIL が観測されている。従って Brown[12] のアドバイスに従い，この FAIL は誤検出であると判断する。

ヒストグラムに問題のある画像 Stream and Bridge について，評価結果を Fig. 13 に示す。64-bit LFSR を使用することにより，Stream and Bridge でも FAIL が抑制できるこ

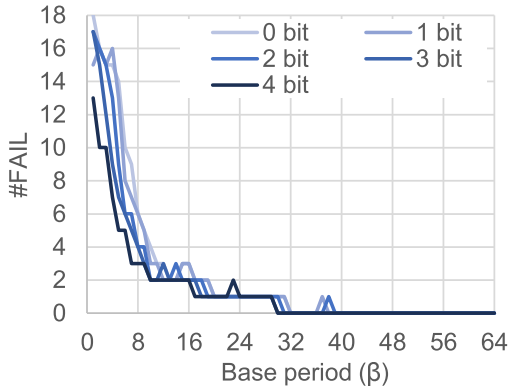

Fig. 10. Diehard test results of the Lenna (64-bit LFSR).

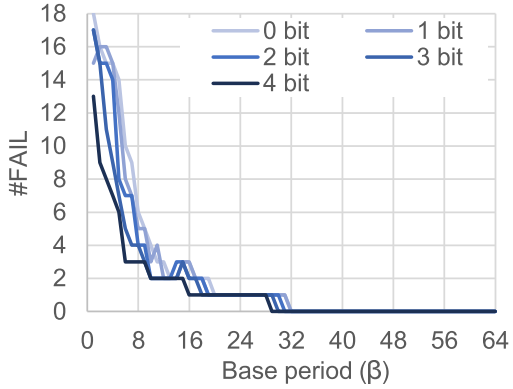

Fig. 11. Diehard test results of the Fishing Boat (64-bit LFSR).

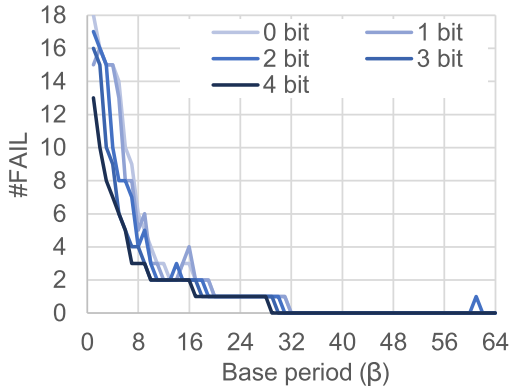

Fig. 12. Diehard test results of the Airport (64-bit LFSR).

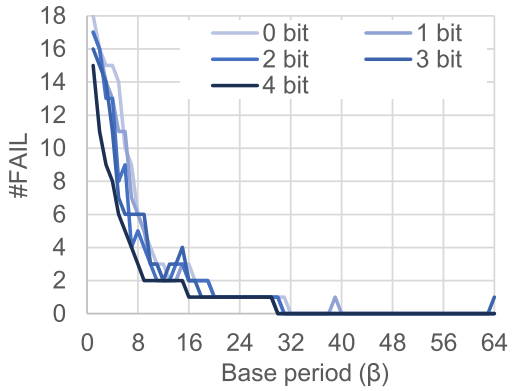

Fig. 13. Diehard test results of the Stream and Bridge (64-bit LFSR).

Table 3. NIST SP 800-22 test results (64-bit LFSR, $\beta = 64$).

Image	$m = 1$	$m = 2$	$m = 3$	$m = 4$
Lenna (256×256)	pass	pass	pass	fail (NonOverlappingTemplate)
Fishing Boat (512×512)	pass	pass	pass	fail (RandomExcursions)
Airport (1024×1024)	pass	pass	pass	fail (RandomExcursions)
Stream and Bridge (512×512)	pass	fail (NonOverlappingTemplate)	pass	pass

とがわかる。$(\beta, m) = (39, 1)$ における FAIL は OPERM5 であり，また $\beta > 39$ では FAIL が発生しないため，実用上の問題はないと思われる。一方，$(\beta, m) = (64, 3)$ における FAIL は Overlapping Sums テストである。近辺の β や他の m では FAIL が発生しておらず，このような孤立的な FAIL の発生について現在まで原因は分かっていない。次章（5 章）でも $m \geq 2$ で乱数品質の低下が検出されており，それと関連した事象であると推測されるが，今のところ詳細は不明である。

5. NIST テストによる評価

Table 3 は，NIST SP 800-22 テストの結果をまとめたものである。〈4·3〉節の検討結果に従い，LFSR 長は 64 bit，$\beta = 64$ とした。合格は pass，不合格は fail と表示し，不合格の場合は失敗したテスト名を括弧内に示した。なお，この条件下では，$m = 0$（揺らぎ無し）でも NIST テストを pass する。

最初に，Lenna，Fishing Boat，Airport の 3 画像に注目する。$m = 1, 2, 3$ における結果は予想通りで，画像サイズによらずテストに合格する。$m = 0$ で実験しても NIST テストに合格することから，64-bit LFSR，$\beta = 64$ という条件下では画像によるエントロピーが小さくても高品質な乱数が生成され，そのため画像サイズが乱数品質に影響を及ぼさないと考えられる。

$m = 4$ においては，いずれの画像でも NIST テストに不合格となっている。この原因については調査中であり，現在のところ詳細不明である。

しかし 2 章でも述べたように，画素値の上位ビットは変化に乏しく，エントロピーは少ないことが予想される。さらに 4 章で見たように，m が大きくなっても乱数品質が向上しないだけでなく，同程度の計算時間で比較すると $m > 1$ で乱数品質が劣化する場合もある。そのため，乱数品質が保てる限り，小さい m を利用することが推奨される。その意味で $m = 4$ の設計には有効性が無く，採用する動機がないため，$m = 4$ における品質低下は問題にならない。

次に，画像 Stream and Bridge の結果を検討する。〈4·2〉節でも述べた通り，本画像には不自然なトーンジャンプがあるため，32-bit LFSR を用いた Diehard テストでは $m = 1, 2$ において乱数品質が低かった。

本章では 64-bit LFSR を用いて評価するため，揺らぎ無し（$m = 0$）でも NIST テストに合格するが，本画像でも $m = 1, 3, 4$ の条件下で NIST テストを pass した。しかし，本画像の下位 2 bit で揺らぎを与えたとき（$m = 2$）は，NIST テストに fail した。

Stream and Bridge は特異なヒストグラムを持つ画像であるため，他の 3 画像とは傾向の異なる結果が得られたと考えられる。全ての特殊画像について乱数品質を保証することは困難であるが，少なくとも今回の実験では 4 画像全てについて $m = 1, 3$ が良好な結果を示している。ただし m が大きくなると計算時間の増大も見込まれることから，乱数品質に問題が無ければ m の値は小さい方が良い。

以上より本研究では，Diehard テストと NIST テストの結果を踏まえて，画素値の最下位ビットを揺らぎとして用いること（$m = 1$）を推奨する。

6. 議　　論

本研究では，乱数品質の評価を（1）Diehard テストによるスクリーニング，（2）NIST SP800-22 テストによる評価，の 2 段階で行っている。Diehard と NIST SP800-22 を採用したのは，これらが広く知られており，多くの先行研究で採用されていることが主な理由である。

本章では，本研究で採用した評価方法の妥当性と，代替手法の可能性について若干の補足を行う。

〈6·1〉 スクリーニング手法　本研究では，妥当な設計パラメータを探索するための簡易的評価手法として Diehard テストを採用している。最終的な品質評価は NIST テストで行うため，少ないデータ量で高速に評価できる Diehard テストを選択した。

Diehard テストには合格判定基準が与えられていないため，本研究では 4 章で述べた基準で結果を判定した。PASS/WEAK/FAIL の 3 段階で評価する手法は，Diehard の拡張版である Dieharder[12] に準じたものである。

Diehard の代わりに Dieharder でスクリーニングを行うことも可能であるが，Dieharder は使用するデータ量や実行時間が大きいため，本研究におけるスクリーニング用途には適していない。Dieharder では，Diehard に新たなテスト項目を加えて拡張しており，最終的には NIST テストの項目すべてを取り込むとされている[13]。このような拡張志向はスクリーニングには不要であるので，本研究では Diehard テストを採用している。

Diehard テストは各項目で 1〜100 個の p 値を出力する。Table 4 は，Diehard の各テストが出力する p 値の数を示したものである。本研究では，〈4·1〉節で述べたように各テストの総合判定を行う。p 値が多いテスト（ケース 1）では，p 値の一様分布を検証するために KS 検定を利用して

Table 4. DIEHARD test[10].

Test	Num. p-values
Birthday Spacings	9
Overlapping 5-permutation (OPERM5)	2
Binary Rank (31x31)	1
Binary Rank (32x32)	1
Binary Rank (6x8)	25
Overlapping 20-tuples Bitstream	20
Overlapping Pairs Sparse Occupancy (OPSO)	23
Overlapping Quads Sparse Occupancy (OQSO)	28
DNA	31
Count the 1's (stream)	1
Count the 1's (specific)	25
Parking Lot	10
Minimum Distance	100
3-d spheres	20
Squeeze	1
Overlapping Sums	10
Runs Up/Down	4
Craps	2
	313

判定する。p 値が少ないテスト（ケース 2）では，1 つでも FAIL 相当の p 値が含まれれば結果は FAIL と判定する。

それぞれの p 値が FAIL と判定される確率を $\epsilon \ll 1$ と書くとき，ケース 2 で一様分布する n 個の p 値が FAIL する確率は $1-(1-\epsilon)^n \approx n\epsilon$ である。即ちケース 2 で FAIL する確率は，p 値の数 n に概ね比例する。従ってケース 2 は比較的小さい n に限定して適用すべきである。テスト毎に FAIL の発生率が大きく異なることは避けたいので，本研究では $n \leq 2$ に対してケース 2 を適用している。このとき Diehard の 18 テストのうち，ケース 1 が 12 テスト，ケース 2 が 6 テストとなる。

一方，ケース 1 は KS 検定で p 値が一様分布しているかを検証しているため，p 値の個数が少ないと結果の信頼性が低くなる。Brown[13] は，KS 検定は 100 個以上の p 値で行うべきであり，20 個程度では充分信頼できないと述べている。Brown のアドバイスを重視するのであれば，各テストで生成する p 値を増やす必要があるが，それでは実行時間が増大しスクリーニングという目的に合わない。ケース 1 の適用範囲を狭める（ケース 2 の適用範囲を広げる）という選択肢もあるが，その場合はケース 2 の範囲内で p 値が多いテストほど FAIL の確率が上がることになり，実質的に p 値の多いテストの重みが増して総合判定の公平性が損なわれる。

本研究では「p 値 3 個以上で KS 検定を適用」しているが，この閾値は絶対的な基準ではない。著者の研究目的に即して，総合判定の公平性を重視しつつ，スクリーニングに適した判定方法を採用した結果である。

〈6･2〉 乱数品質の評価手法 最終評価に使用している NIST テストも，完璧なものとは言えない。これまでも 2001 年，2008 年，2010 年と改訂されており，現在も再改訂の作業中である[14]。こうした改訂が乱数検定結果に及ぼす影響については，今後の課題とする。

本研究では NIST テストで乱数品質の検証を行ったが，他の乱数検定テストの使用を否定するものではない。NIST SP800-22[7] の Abstract にも書かれている通り，統計的乱数検定スイートは乱数生成器が特定応用に使用可能であるか調べるための最初の道具であり，どのような検定スイートも絶対的な保証を与えることはできない。本研究では NIST テストを採用したが，ユーザは必要に応じて他の乱数検定テストを利用し，個別の技術的要求に従った品質評価を行うことが推奨される。

本研究では画像データと LFSR を用いた URNG を提案し，その URNG が現状の NIST テストに合格することを示したが，あらゆる状況下で高品質な乱数が生成されるとは保証していない。出力される乱数の品質は，LFSR の特性多項式の選択，LFSR の初期値，入力画像の性質，など多くの要素によって影響を受ける。本研究や先行研究で幾つかの設計パラメータについて品質評価を行っているが，URNG を使用する際には各自の応用や設計に合わせた品質評価を行うべきである。

7. おわりに

本研究の実験結果より，64-bit LFSR を使用し，充分大きな基本サンプリング間隔 $\beta \geq 40$ で使用することにより，Diehard テストに合格する乱数列が生成できることが分かった。画像サイズはモノクロ 256×256 でも Diehard テストに合格するが，512×512 あるいは 1024×1024 であれば乱数品質が向上する可能性がある。

また，乱数品質の観点からは，揺らぎ $\alpha(n)$ に画素値の最下位 1 ビットを用いれば十分であり，下位複数ビットを使用しても計算時間増大に対して乱数品質の向上は期待できないことが観察された。そのため本研究では，画素値の最下位ビットを揺らぎとして用いることを推奨する。

これらの知見を踏まえて，提案手法の乱数品質を NIST SP 800-22 テストで検証した。64-bit LFSR，基本間隔 $\beta = 64$ という条件下で，提案手法の生成した乱数列は NIST テストに合格した。

本研究の提案手法を利用すれば，カメラ付き携帯端末など撮像機能のある多くのデバイスにおいて，高品質かつ予測不能な乱数列を生成することができる。必要な計算量や記憶容量も少なく，処理能力の限られた組込みデバイスでも利用可能な乱数生成手法といえる。

謝 辞

本研究の一部は JSPS 科研費 20K11733 の支援による。

文 献

（1） A. Suciu, S. Banescu, and K. Marton: "Unpredictable random number generator based on hardware performance counters", Digital Information Processing and Communications (ICDIPC 2011), pp.123–137, Springer-Verlag (2011)

（2） K. Marton, A. Zaharia, S. Banescu, and A. Suciu: "Randomness Assessment of an Unpredictable Random Number Generator based on Hardware Performance Counters", *Romanian Journal of Information Science and Technology*, Vol.20, No.2, pp.136–160 (2017)

（3） H. Masaoka, S. Ichikawa, and N. Fujieda: “Random Number Generation from Internal LFSR and Fluctuation of Sampling Interval”, *IEEJ Trans. Industry Applications*, Vol.141, No.2, pp.86–92 (2021) (in Japanese)
正岡秀崇・市川周一・藤枝直輝：「内蔵 LFSR とサンプリング間隔の揺らぎを利用した乱数生成手法」, 電学論 D, Vol.141, No.2, pp.86–92 (2021)
（4） H. Kamogari and S. Ichikawa: “Evaluation of a random number generator based on an internal linear feedback shift register”, *IEEJ Trans. Industry Applications*, Vol.143, No.2, pp.87–93 (2023) (in Japanese)
鴨狩滉斗・市川周一：「内蔵 LFSR を用いた乱数生成方法の評価」, 電学論 D, Vol.143, No.2, pp.87–93 (2023)
（5） N. Fernández, F. Quintas, L. Sánchez, and J. Arias: “Social noise: Generating random numbers from twitter streams”, *Fluctuation and Noise Letters*, Vol.14, No.1, art. No.1550012 (2015)
（6） A. Chiba and S. Ichikawa: “Evaluation of Random Number Generator Utilizing Weather Data and LFSR”, *IEEJ Trans. Industry Applications*, Vol.143, No.2, pp.80–86 (2023) (in Japanese)
千葉歩武・市川周一：「気象データと LFSR による乱数生成手法の評価」, 電学論 D, Vol.143, No.2, pp.80–86 (2023)
（7） A. Rukhin, et al.: “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”, NIST SP 800-22 (Rev. la) (2010)
（8） T. Betchaku and S. Ichikawa: “Improvement of the URNG that utilizes weather data and LFSR”, The Papers of Technical Meeting on Innovative Industrial System, IEE Japan, IIS-23-014 (2023) (in Japanese)
別役拓哉・市川周一：「気象データと LFSR を用いた URNG の改良」, 電学次世代産業システム研, IIS-23-014 (2023)
（9） S. Ichikawa: “Random Number Generation from Image Data and LFSR: a Preliminary Study”, The Papers of Technical Meeting on Innovative Industrial System, IEE Japan, IIS-23-056 (2023) (in Japanese)
市川周一：「画像データと LFSR による乱数生成手法の検討」, 電学次世代産業システム研, IIS-23-056 (2023)
（10） G. Marsaglia: “Diehard battery of tests of randomness (Archived)”, https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/
（11） J. Rajski and J. Tyszer: “Primitive Polynomials Over GF(2) of Degree up to 660 with Uniformly Distributed Coefficients”, *Journal of Electronic Testing*, Vol.19, pp.645–657 (2003)
（12） R.G. Brown: “Robert G. Brown’s General Tools Page”, https://webhome.phy.duke.edu/~rgb/General/dieharder.php (accessed 8 April 2024)
（13） R.G. Brown: “DieHarder: A Gnu Public License Random Number Tester”, https://rurban.github.io/dieharder/manual/dieharder.pdf (accessed 8 April 2024)
（14） NIST: “Decision to Revise NIST SP 800-22 Rev. 1a”, https://www.nist.gov/news-events/news/2022/04/decision-revise-nist-sp-800-22-rev-1a (accessed 8 April 2024)

市 川 周 一（上級会員） 1985 年東京大学理学部卒業。1987 年同大学大学院理学系研究科修士課程修了。1987 年新技術事業団，1991 年三菱電機（株），1994 年名古屋大学工学部助手。1997 年豊橋技術科学大学工学部講師。同助教授，准教授を経て，2011 年沼津工業高等専門学校制御情報工学科教授。2012 年より豊橋技術科学大学大学院工学研究科教授。現在に至る。理学博士。IEEE (senior member)，電子情報通信学会（シニア会員），ACM，情報処理学会，各会員。