Pseudo-Random Number Generation by Staggered Sampling of LFSR

Shuichi Ichikawa
Department of Electrical and Electronic Information Engineering
Toyohashi University of Technology
Toyohashi 441-8580, Japan
ichikawa@ieee.org

***Abstract*—Linear Feedback Shift Register (LFSR) is widely used as a simple Pseudo-Random Number Generator (PRNG). In 2009, Gu and Zhang proposed a Leap-ahead LFSR, which applies the feedback polynomial multiple times to achieve a larger generation rate. Though a Leap-ahead LFSR applies the feedback polynomial a fixed number of times, the quality of randomness could be potentially improved by applying the feedback polynomial variable times. This study introduces a Staggered LFSR, where a subordinate LFSR determines the number of feedback polynomial applications for the main LFSR. Our evaluation results demonstrate that a Staggered LFSR exhibits better randomness quality compared to a Leap-ahead LFSR of the same length. Though the subordinate LFSR requires an additional cost, the overhead is minimal. The Staggered LFSR, designed with appropriate parameters, successfully passed both the Diehard test and the NIST SP 800-22 test.**

***Index Terms*—Random number, Linear Feedback Shift Register, randomness test**

I. Introduction

Random number generation is an essential part of many applications (e.g., simulations). Random numbers are usually categorized into two kinds: true random numbers (TRN) and pseudo-random numbers (PRN). TRN is generated by physical phenomena such as thermal noise and is hence unpredictable. On the other hand, PRN is numerically generated by deterministic algorithms. This study focuses on PRN generation methods.

Linear Feedback Shift Register (LFSR) is popular as a Pseudo-Random Number Generator (PRNG) and widely adopted in both software and hardware due to its simplicity and ease of implementation.

LFSR generates only one random bit per cycle, which may be insufficient for applications requiring a large number of random numbers. To increase the generation rate, it is possible to use two or more LFSRs in parallel. However, this introduces a correlation between the LFSRs, which becomes a new problem. Additionally, the parallel use of LFSRs requires more hardware resources.

Gu and Zhang [1] proposed the Leap-ahead LFSR, which enhances the generation rate of random bits by applying the feedback polynomial multiple times in each cycle. This concept is similar to loop unrolling [2] in software development. Lee and Kim [3] introduced a segmented Leap-ahead LFSR to mitigate the correlation in the output stream of a Leap-ahead LFSR. They also reported an extended period without compromising area and performance overhead. Tan et al. [4] reported that a TL LFSR, which is a PRNG based on the Leap-ahead LFSR, successfully passed the NIST test of randomness.

This study introduces a new PRNG called Staggered LFSR, which samples the value of the LFSR at variable periods. Staggered LFSR is based on the Leap-ahead LFSR concept, with the distinction that the feedback polynomial is applied a variable number of times. While preceding studies lacked sufficient results regarding the quality of randomness, this study quantitatively investigates the relationship between design parameters and the quality of randomness across a wide variety of designs. The evaluation results of this work are expected to provide valuable insights for numerous applications.

II. Background

Masaoka, Ichikawa, and Fujieda [5] enhanced an embedded processor by adding a 128-bit LFSR and utilized it to generate random number sequences. They achieved this by sampling the lower-most word (32 bits) at uncertain intervals. This idea resembles to the Leap-ahead LFSR, but it generates practically unpredictable random numbers due to its uncertainty of sampling intervals. Consequently, their method is classified as a URNG (Unpredictable Random Number Generator) [6] [7] rather than a PRNG. The entropy source for their URNG relies on the fluctuations in sampling intervals, which are generated by external interrupts and the activities of other processes within the system. Masaoka et al. [5] implemented their system on an FPGA board and reported successful results in a randomness test (Diehard test [8]).

Kamogari and Ichikawa [9] extensively discussed the design parameters of the URNG introduced by Masaoka et al. [5] through comprehensive simulations. They systematically investigated the randomness qualities across various combinations of LFSR and sampling intervals, aiming to identify the URNG specifications that successfully pass randomness tests.

In their study, Kamogari and Ichikawa [9] examined the randomness quality by introducing various fluctuations into the sampling intervals. Notably, their results without any fluctuations correspond to the evaluation of the Leap-ahead LFSR. Hence, the findings presented by Kamogari and Ichikawa [9] encompass the randomness evaluations of various Leap-ahead LFSRs.

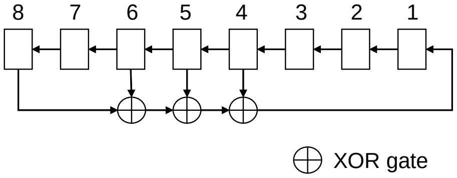

Fig. 1. An example of 8-bit LFSR [8,6,5,4] [9]

The URNG proposed by Masaoka et al. [5] relies on a hardware LFSR and the fluctuations in sampling intervals caused by external factors. However, it is possible to utilize alternative entropy sources to introduce fluctuations in the sampling intervals, enabling the implementation of the URNG without hardware assistance.

Chiba and Ichikawa [11] suggested using publicly available weather data from the Internet as an entropy source to implement the URNG with a software LFSR. Since weather data originate from natural phenomena, the resulting random numbers exhibit practical unpredictability. Chiba and Ichikawa utilized wind direction data from 25 observation points spanning 10 years and reported successful results in passing the NIST test [10] with the generated random numbers. Furthermore, Betchaku and Ichikawa [12] demonstrated that the volume of weather data can be reduced by employing advanced hash functions and optimizing the data order.

The present study addresses the following questions: Can artificial data be utilized as an entropy source to introduce fluctuations? Specifically, can an LFSR be employed to generate fluctuations in sampling intervals that successfully pass randomness tests?

When utilizing an LFSR to fluctuate the sampling intervals, the resulting output data are solely determined by the internal states of the RNG. Consequently, this RNG is classified as a PRNG. Specifically, this PRNG, which samples an LFSR at variable intervals, can be seen as an enhancement of the Leap-ahead LFSR that samples an LFSR at a fixed interval. Although this study shares a common aspect with previous works [1] [3] [4], the enhancement proposed in this study differs from those in the preceding studies [3] [4]. Furthermore, our method represents a generalization that encompasses the Leap-ahead LFSR as a special case.

Another notable advantage of this work is the presentation of a wide range of randomness quality results for various sets of design parameters. In contrast, previous studies primarily focused on evaluating logic scale for hardware implementation. By providing comprehensive data, this study offers valuable insights for numerous applications, enhancing its practical relevance.

III. Staggered LFSR

This study presents a new random number generator that utilizes an LFSR, following the preceding studies stated in the Section II.

TABLE I
Tap sequences adopted in this study

Length	Tap sequence	Source
8	[8, 7, 2, 1]	Živković [13]
10	[10, 8, 3, 2]	
12	[12, 10, 2, 1]	
14	[14, 12, 11, 1]	
16	[16, 15, 12, 10]	
32	[32, 25, 15, 7]	Rajski [14]
36	[36, 25, 17, 8]	
40	[40, 29, 21, 10]	
44	[44, 31, 22, 11]	
48	[48, 38, 26, 13]	
64	[64, 45, 31, 14]	

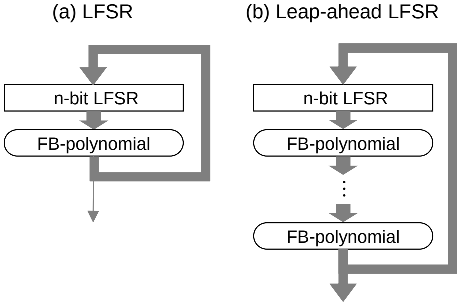

Fig. 2. Leap-ahead LFSR

LFSR is represented by the tap sequence, which corresponds to its feedback polynomial. Figure 1 illustrates an example of 8-bit LFSR, whose tap sequence is [8, 6, 5, 4]. If the feedback polynomial is chosen to be primitive, the period of n-bit LFSR becomes $2^n - 1$. The tap sequences of the LFSRs adopted in the subsequent discussions are summarized in Table I, based on the findings of Kamogari and Ichikawa [9].

A standard LFSR (Fig. 2(a)) applies its feedback polynomial once during each cycle to update its internal state and outputs a single bit. On the other hand, a Leap-ahead LFSR (Fig. 2(b)) applies its feedback polynomial multiple times during each cycle, resulting in multiple bits being generated to increase the output rate [1]. In this study, if the width $n \geq 32$, the lower-most 32 bits are output.

Though a Leap-ahead LFSR applies its feedback polynomial a fixed number of times within a cycle, the proposed method (Staggered LFSR) applies it a variable number of times, allowing for possible fluctuations in the number of applications. The Staggered LFSR is illustrated in Fig.3. It is important to note that a Staggered LFSR is a natural enhancement of a Leap-ahead LFSR. As in the Leap-ahead LFSR, the n-bit Staggered LFSR outputs the lower-most 32 bits when $n \geq 32$.

Kamogari and Ichikawa [9] utilized a high-quality PRNG, known as Mersenne twister [15], to generate the fluctuations in their simulations. Chiba and Ichikawa [11] employed natural entropy derived from weather data to determine the fluctuations. The present study investigates the potential for generating high-quality random numbers using a simpler mechanism by employing a small LFSR to generate fluctuations.

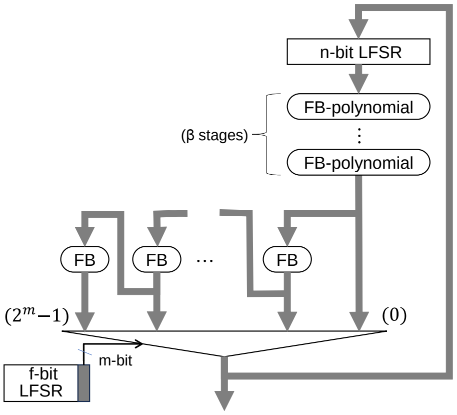

Fig. 3. Staggered LFSR

In the following descriptions, k denotes the index of iteration. After the $(k-1)$-th output, the feedback polynomial is applied $S(k)$ times before the k-th output is sampled. The value of $S(k)$ is determined by the following equations.

$$\begin{aligned} S(k) &= \alpha(k) + \beta && (1) \\ \alpha(k) &= LFSR_f(k) \wedge (2^m - 1) && (2) \end{aligned}$$

Here, the constant β represents the base interval (or base period), and $\alpha(k)$ represents the fluctuation of intervals. $LFSR_f(k)$ refers to the value of the f-bit LFSR at the k-th cycle. The f-bit LFSR is updated simultaneously with the n-bit LFSR.

Equation 2 signifies that the lower-most m bits of the f-bit LFSR are utilized as the fluctuation (Fig. 3). When m is zero, $\alpha(k)$ also becomes 0, leading to $S(k) = \beta$ (a constant). Therefore, the case of $m = 0$ corresponds to the Leap-ahead LFSR. Consequently, the Staggered LFSR can be seen as a natural enhancement of the Leap-ahead LFSR.

In a software implementation, the computation time increases proportionally to $S(k)$, as the feedback polynomial is applied $S(k)$ times. In a hardware implementation, the logic scale and latency of the circuit increase with $S(k)$. In both cases, an increase in $S(k)$ has a negative impact. Therefore, it is crucial to carefully consider the trade-offs between the quality of randomness and the value of $S(k)$. In this study, evaluations will be conducted for values of $0 \leq m \leq 4$ and $1 \leq \beta \leq 64$.

It is crucial to consider the period when designing a PRNG, as a short period can lead to a degradation in the quality of randomness. Since the period is influenced by the combination of design parameters, the designer must make careful selections to avoid a short period.

The period A of a Leap-ahead LFSR can be expressed using the following equation [1]:

$$A = \frac{LCM(2^n - 1, \beta)}{\beta} \qquad (3)$$

In the equation, the variable n represents the length of the LFSR, while the constant β corresponds to the number of applications of the feedback polynomial, which can also be interpreted as the degree of unrolling. The function $LCM(x, y)$ denotes the least common multiple of x and y.

TABLE II
DIEHARD EVALUATION CRITERIA [5]

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

To maximize the period of a Leap-ahead LFSR, two integers $2^n - 1$ and β have to be relatively prime. In other words, the designer should select the pair (n, β) in such a way that the greatest common divisor $GCD(2^n - 1, \beta)$ is minimized.

The period P of the Staggered LFSR can be calculated using the following formula, where B represents the sum of fluctuations within a period of f-bit LFSR:

$$\begin{aligned} B &= \sum_{k=1}^{2^f-1} S(k) = \sum_{k=1}^{2^f-1} \alpha(k) + \sum_{k=1}^{2^f-1} \beta \\ &= 2^{f-m} \cdot \frac{2^m(2^m-1)}{2} + (2^f - 1)\beta \\ &= 2^{f-1}(2^m - 1) + (2^f - 1)\beta && (4) \\ P &= \frac{LCM(2^n - 1, B)}{B} && (5) \end{aligned}$$

In the case of the Staggered LFSR, the designer should choose the parameter set (f, m, n, β) in a manner that minimizes the greatest common divisor $GCD(2^n - 1, B)$.

It is important to note that the period alone does not determine the randomness of the generated sequence. The quality of randomness should be evaluated through appropriate testing of the generated random sequence. Therefore, the designer should carefully choose the parameter set, as a short period can potentially result in a degradation of randomness. However, it is possible for a non-maximum period to be sufficient if the output data satisfies the required criteria, such as passing specific randomness tests.

IV. Evaluation with the Diehard test

Randomness tests typically require a substantial amount of data to ensure reliable evaluation. For instance, the Diehard test [8] recommends a dataset size of approximately 10^8 bits, while the NIST test [10] suggests testing 1000 sets of 10^6 bits each, resulting in a total of 10^9 bits.

In this section, the Diehard test [8] is adopted, as in previous studies [5] [9] [11]. The Diehard test is comparatively less stringent compared to the NIST test, but it requires less time for each trial. As a result, it is well-suited for the development stage where numerous trials are necessary.

A. Interpretation

The Diehard test comprises 18 individual tests, each generating 1 to 100 p-values (313 p-values in total). The interpretation of the test results is not standardized and is left to the

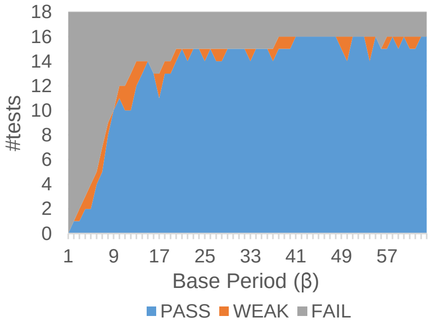

Fig. 4. Diehard test results of the 32-bit Leap-ahead LFSRs.

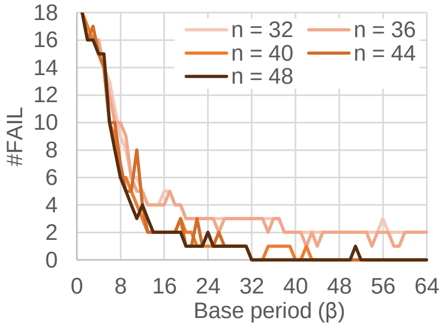

Fig. 5. Diehard test results of the Leap-ahead LFSRs.

discretion of each user. In this study, the evaluation criteria adopted in Masaoka et al. [5] are adopted.

The interpretation of each p-value is based on the assumption that the p-values should be uniformly distributed in the interval [0, 1). The results of each p-value are interpreted according to Table II. It is expected that the probability of observing a FAIL is 2×10^{-6} under the assumption that the input data is random. Therefore, if the input data is truly random, the occurrence of FAIL should be practically nonexistent. Similarly, the expected probability of observing a WEAK result is 1×10^{-2}, indicating that there is no cause for concern if WEAK results are observed in approximately 1% of the p-values.

To summarize the interpretation of the 313 p-values generated by the Diehard test, the following rules are applied:

- If there are three or more p-values:
  - The uniformity of the p-values is assessed using the Kolmogorov-Smirnov (KS) test.
  - The result of the KS test is interpreted according to the guidelines presented in Table II.
- If there are one or two p-values:
  - If any of the p-values are interpreted as FAIL, the overall decision is FAIL.
  - Else if any of the p-values are interpreted as PASS, the overall decision is PASS.
  - Else the overall decision is WEAK.

The quality of randomness is determined based on the collective results obtained from the 18 individual tests conducted in the Diehard test, following the aforementioned interpretation rules.

B. Leap-ahead LFSR

Figure 4 provides a visual representation of the Diehard test results obtained for the 32-bit Leap-ahead LFSR. The base period β is depicted along the horizontal axis, while the vertical axis displays the number of tests categorized based on the decisions of the 18 individual tests.

Despite the improvement in randomness quality as β increases, it should be noted that the 32-bit Leap-ahead LFSR is unable to pass all individual tests. Specifically, two tests, namely the Binary Rank (31x31) test and the Binary Rank (32x32) test, consistently failed for all values of β within the range of $38 \leq \beta \leq 64$. These Rank tests require a longer period than the period of $2^{32} - 1$ offered by the 32-bit LFSR, leading to their failure. Furthermore, the Overlapping Sums test also failed at $\beta = 56$.

To further investigate the effectiveness of Leap-ahead LFSRs with lengths beyond $n = 32$, we focus on the number of FAILs for simplicity and ease of understanding.

Figure 5 provides a summary of the number of FAILs observed for Leap-ahead LFSRs with various lengths, including $n = 32, 36, 40, 44,$ and 48. For the case of $n = 40$, no FAILs are observed for $\beta \geq 43$. The only FAIL observed at $(n, \beta) = (40, 42)$ occurred in the OPERM5 test. It is worth noting that Brown [16] suggested a bug in the OPERM5 test, and Kamogari and Ichikawa [9] also reported peculiar FAILs in the OPERM5 test. Following Brown's advice [16], this particular FAIL is disregarded as a false positive of the OPERM5 test.

Based on this decision, the design with $n = 40$ successfully passes the Diehard test for $\beta \geq 40$, while the designs with $n = 44$ and $n = 48$ pass for $\beta \geq 32$. The fail at $(n, \beta) = (48, 51)$, which was caused by OPERM5, is also regarded as a false positive.

C. 32-bit Staggered LFSR

This section presents the Diehard test results for the 32-bit Staggered LFSR ($n = 32$) with different values of f and m.

Figure 6 illustrates the number of FAILs for the Staggered LFSR with $f = 12, 14, 16$, where the least significant bit is used for fluctuations ($m = 1$). In the graph, the acronym L.A. designates the results of the 32-bit Leap-ahead LFSR with no fluctuation.

The 32-bit Leap-ahead LFSR fails to suppress FAILs for $1 \leq \beta \leq 64$, as mentioned in Section IV-B. Similarly, the 32-bit Staggered LFSR with $f = 10$ also fails for $1 \leq \beta \leq 64$.

By increasing the value of f, the 32-bit Staggered LFSR exhibits significant improvement in its ability to suppress FAILs. Specifically, no fails occur with $f = 12$ for $\beta \geq 62$, $f = 14$ for $\beta \geq 42$, and $f = 16$ for $\beta \geq 38$. In comparison to the 32-bit Leap-ahead LFSR, which fails to pass the Diehard test, the 32-bit Staggered LFSR demonstrates superior quality as it successfully passes the Diehard test with $f \geq 12$.

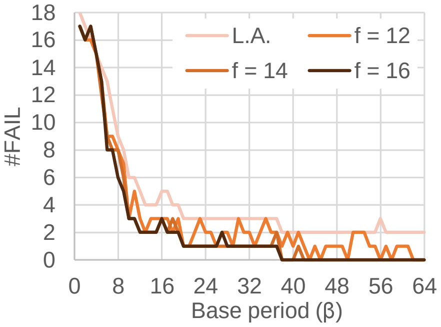

Fig. 6. Diehard test results of the 32-bit Staggered LFSR (m = 1).

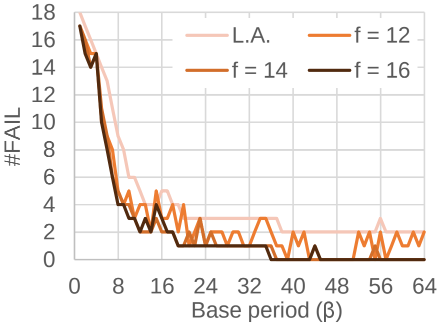

Fig. 7. Diehard test results of the 32-bit Staggered LFSR (m = 2).

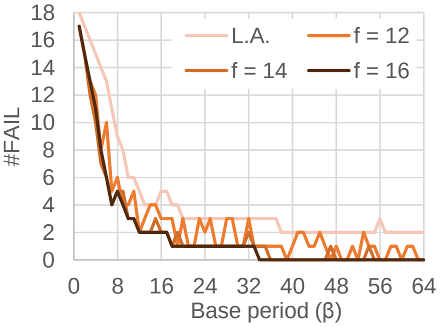

Fig. 8. Diehard test results of the 32-bit Staggered LFSR (m = 3).

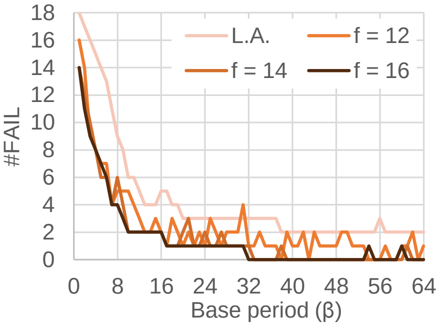

Fig. 9. Diehard test results of the 32-bit Staggered LFSR (m = 4).

Furthermore, Figures 7, 8, and 9 present the Diehard test results for $m = 2, 3, 4$, respectively, similar to Figure 6 where $m = 1$. It is observed that for $m = 2, 3, 4$, no fails occur for large values of β with $f = 14$ and $f = 16$. Therefore, we can conclude that the proposed design has improved the randomness quality.

For larger values of m, the number of FAILs may appear smaller for the same β. This should be interpreted as follows: Let $\bar{\alpha}$ denote the average value of $\alpha(k)$. For $m = 1, 2, 3, 4$, the corresponding values of $\bar{\alpha}$ are $0.5, 1.5, 3.5, 7.5$. Since the horizontal axes of Figures 6 to 9 represent β, the number of applications of the feedback polynomial at the same β increases by $\bar{\alpha}$ as m increases. This partially explains the observed appearance of smaller FAILs for larger m values.

It has been observed that when m is large, some tests fail with a large value of β. For instance, the Binary Rank (32x32) test fails with $(m, f, \beta) = (4, 14, 61)$, while other values of β around 61 do not exhibit any failures. Similarly, the Overlapping Sums test fails with $(m, f, \beta) = (4, 16, 60)$, while other values of β around 60 do not indicate any failures. Such sporadic failures are not observed when $m = 1$.

Furthermore, designs with $m > 1$ result in an increase in average calculation time in software implementation and also lead to larger logic scale and latency in hardware implementation, as illustrated in Figure 3.

Considering all these factors, it is strongly recommended to utilize the Staggered LFSR with the parameter $m = 1$. This choice not only avoids the occurrence of sporadic failures but also helps minimize the negative impact on the design, making it a practical and efficient option.

D. 36-bit Staggered LFSR

Figures 10 to 13 provide a summary of the number of FAILs for the 36-bit Staggered LFSRs with different values of f and m, compared to the performance of the 36-bit Leap-ahead LFSR (L.A.).

The 36-bit Leap-ahead LFSR fails to pass the Diehard test within the range of $1 \leq \beta \leq 64$. In contrast, the 36-bit Staggered LFSR, particularly with $f = 8$ and $f = 10$, manages to suppress the number of FAILs significantly. Moreover, for $f = 12$ and $\beta \geq 32$, the 36-bit Staggered LFSR successfully passes the Diehard test for all m values ranging from 1 to 4.

In the case of $n = 32$, the Staggered LFSR requires $f \geq 14$ and $\beta \geq 42$ for stable success in the Diehard test. However, the 36-bit Staggered LFSR achieves a passing result in the Diehard test with $f \geq 12$ and $\beta \geq 38$.

These observations indicate a tradeoff between the two parameters n and f in order to achieve success in the Diehard test. On the other hand, the n-bit Leap-ahead LFSR successfully passes the Diehard test with $n \geq 44$ and $\beta \geq 32$ without any fluctuations, which corresponds to $f = 0$ of the Staggered LFSR.

V. Evaluation with the NIST test

In this section, the design parameters discussed in Section IV are examined using the NIST SP 800-22 test [10].

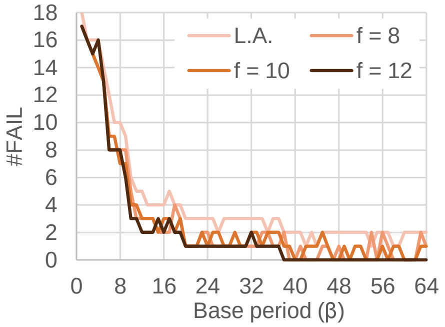

Fig. 10. Diehard test results of the 36-bit Staggered LFSR (m = 1).

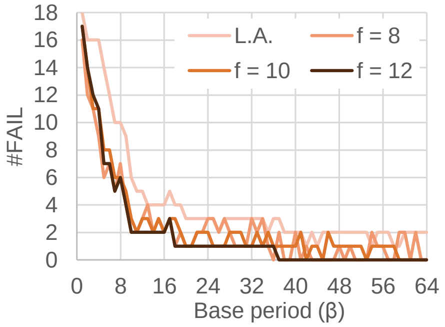

Fig. 12. Diehard test results of the 36-bit Staggered LFSR (m = 3).

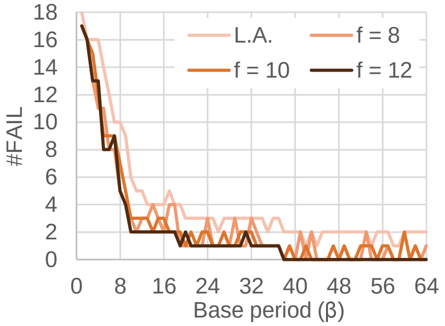

Fig. 11. Diehard test results of the 36-bit Staggered LFSR (m = 2).

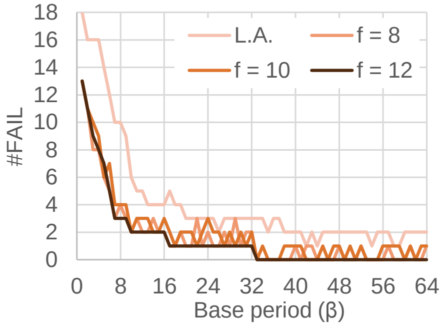

Fig. 13. Diehard test results of the 36-bit Staggered LFSR (m = 4).

Table III presents the results of the NIST test for n-bit Leap-ahead LFSRs with different lengths ($n = 32$, ..., 64). The base period β is set to 64, as in Figure 5 of the Diehard test.

The "Result" column in the table indicates the overall test result, classified as either *pass* or *fail*, based on the specifications provided by NIST. In the case of a *fail*, the individual tests that failed are listed in parentheses.

Similar to the results obtained from the Diehard test, the Leap-ahead LFSRs with shorter periods (i.e., smaller n) tend to fail the NIST test, while those with longer periods (i.e., larger n) pass the NIST test. This consistency between two tests support the validity of our approach.

The design $(n, \beta) = (40, 64)$ passes the Diehard test (Fig. 5), while it fails NonOverlappingTemplate test in the NIST test. This observation suggests that designs that are near the borderline, such as the $(40, 64)$ design, may perform well in less stringent tests like the Diehard test, but they may not meet the more rigorous requirements of the NIST test.

Table IV lists the results of the Staggered LFSR for various combinations of n, f, and m, which are considered to reside near the borderline.

The design $(n, f, \beta) = (32, 14, 64)$ fails the NIST test, though it successfully passes the Diehard test for $1 \leq m \leq 4$ (Figures 6 to 9). The design $(n, f, \beta) = (32, 16, 64)$, where the fluctuation LFSR is 2 bits longer, successfully passes the NIST test for $m = 1, 2, 4$ and it also passes the Diehard test for $m = 1, 2, 3, 4$. These results are consistent with the fact that the NIST test is more stringent than the Diehard test.

The design $(n, f, \beta) = (36, 10, 64)$ successfully passes the Diehard test for $m = 2, 3$, but it fails for $m = 1, 4$ due to the Binary Rank (31x31) test. Similarly, it passes the NIST test for $m = 2, 4$, but fails for $m = 1, 3$. Considering the overall results, the design $(36, 10, 64)$ is not considered qualified due to a significant number of failures across different design parameters. However, by increasing the length of f by 2 bits, the design $(n, f, \beta) = (36, 12, 64)$ successfully passes the Diehard test for $m = 1, 2, 3$, as well as the NIST test for $m = 1, 2, 4$.

It is worth noting that designs such as $(32, 16, 3)$ and $(36, 12, 4)$ exhibit sporadic failures, where some values of m lead to a *fail* while others result in a *pass*. This observed pattern of sporadic failures agrees with the findings reported in Section IV-C for large m values in the Diehard test. The underlying reason or mechanism behind these sporadic failures has yet to be fully elucidated.

However, this peculiar behavior is not observed in the case of $m = 1$ in either the Diehard test or the NIST test. As mentioned in Section IV-C, large values of m have a negative impact on the implementation, and now the quality issue associated with large m values has been added. Therefore, it is natural to recommend avoiding designs with $m \geq 2$.

In summary, this study recommends utilizing the least significant bit ($m = 1$) of the f-bit LFSR when designing a Staggered LFSR, as it offers a combination of good randomness quality and lower implementation cost. For practical applications, Table IV includes the evaluation results of

TABLE III
NIST TEST RESULTS OF n-BIT LEAP-AHEAD LFSR ($\beta = 64$)

n	Result
32	fail (Rank)
36	fail (Rank, NonOverlappingTemplate)
40	fail (NonOverlappingTemplate)
44	pass
48	pass
64	pass

TABLE IV
NIST TEST RESULTS OF STAGGERED LFSR ($\beta = 64$)

n	f	m	Result
32	14	1	fail (Rank, RandomExcursions)
		2	fail (Rank, LongestRun, NonOverlappingTemplate)
		3	fail (Rank)
		4	fail (Rank)
	16	1	pass
		2	pass
		3	fail (Rank, NonOverlappingTemplate)
		4	pass
36	10	1	fail (Universal)
		2	pass
		3	fail (Rank)
		4	pass
	12	1	pass
		2	pass
		3	pass
		4	fail (Rank)
64	4	1	pass
	8		pass
	12		pass
	16		pass
	20		pass

$n = 64, m = 1$ for various f.

VI. CONCLUSION

The findings of this study demonstrate that the Staggered LFSR exhibits superior randomness quality compared to a Leap-ahead LFSR of the same length, n. However, the study also identified the requirement for the length, f, of the LFSR for fluctuations. The results of the Diehard and NIST tests indicate that $n+f \gtrsim 48$ is necessary. Although the requirement for the Staggered LFSR (48 bits) is slightly larger than the 44-bit Leap-ahead LFSR, which passes the NIST test, it does not pose significant issues since the implementation cost of an LFSR is relatively low.

Based on our results, the design with a small fluctuation ($m = 1$, $\bar{\alpha} = 0.5$) provides sufficiently high randomness. By adopting the design $m = 1$, the increase in cost for the Staggered LFSR can be minimized compared to the Leap-ahead LFSR. Furthermore, our findings reveal that designs with $m > 2$ sometimes experience a degradation in randomness quality. Considering these observations, this study recommends adopting $m = 1$ for the design of the Staggered LFSR.

The Staggered LFSR offers a wider range of design options compared to the Leap-ahead LFSR, providing users with larger flexibility in choosing the most suitable design for specific applications. The evaluation results of this study are also expected to serve as the foundation for designing the URNG with LFSR [5] [11].

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 20K11733.

REFERENCES

[1] X.C. Gu, M.X. Zhang, "Uniform Random Number Generator Using Leap Ahead LFSR Architecture," Proc. 2009 International Conference on Computer and Communications Security, Dec. 2009, pp. 150-154.

[2] K.R. Wadleigh, I.L. Crawford, "Software Optimization for High-Performance Computing," Prentice Hall, Jan. 2000.

[3] J.H. Lee, S.K. Kim, "Segmented Leap-Ahead LFSR Architecture for Uniform Random Number Generator," International Journal of Software Engineering and Its Applications, vol.7, no.5, pp.233-242, 2013.

[4] Z. Tan, W. Guo, G. Gong, H. Lu, "A New Pseudo-Random Number Generator Based on the Leap-Ahead LFSR Architecture," Proc. 2018 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA), pp. 57-58, Nov. 2018.

[5] H. Masaoka, S. Ichikawa, N. Fujieda, "Random Number Generation from Internal LFSR and Fluctuation of Sampling Interval," IEEJ Trans. Industry Applications, vol. 141, no. 2, pp. 86-92, Feb. 2021. (in Japanese)

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

[7] K. Marton, A. Zaharia, S. Banescu, 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.

[8] G. Marsaglia, "Diehard battery of tests of randomness (Archived)," https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/

[9] H. Kamogari, 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, Feb. 2023. (in Japanese)

[10] A. Rukhin et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST SP 800-22 (Rev. la), Apr. 2010.

[11] A. Chiba, S. Ichikawa, "Evaluation of Random Number Generator Utilizing Weather Data and LFSR," IEEJ Trans. Industry Applications, vol. 143, no. 2, pp. 80-86, Feb. 2023. (in Japanese)

[12] T. Betchaku, 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, Mar. 2023. (in Japanese)

[13] M. Živković, "A table of primitive binary polynomials," Mathematics of Computation, vol. 62, no. 205, pp. 385-386, Jan. 1994.

[14] J. Rajski, 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.

[15] M. Matsumoto, T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Trans. Modeling and Computer Simulation, vol. 8, no. 1, pp. 3–30, Jan. 1998.

[16] R. G. Brown, "Robert G. Brown's General Tools Page," https://webhome.phy.duke.edu/~rgb/General/dieharder.php (accessed 26 Jun. 2023)