

Comparative Study of Pseudo-Noise (PN) Sequence Generator

Tanish Narendra Vatpal¹, Krishna Ishwar Pawar², Sneha Nitin Andhalkar³, Avinash Suresh Gholap⁴

Department of Electronics and Telecommunication Engineering, MVPS's KBT College of Engineering, Nashik, Maharashtra, India

Abstract— Pseudo-noise (PN) sequences are deterministic binary sequences which have some of the desired statistical characteristics of random noise but can be exactly reproduced. They are significant for spread-spectrum communications, synchronization, range finding, scrambling, and channel testing and digital-system verification. This undergraduate research paper describes the design, mathematical foundation, procedure of implementation and performance analysis of LFSR based PN sequence generator. The study covers finite-field representation, primitive feedback polynomials, maximal-length sequences, seed selection, state transitions, balance, run distribution, autocorrelation and implementation constraints. A four stage example is detailed as an illustrative case followed by an evaluation methodology to be adapted for implementation in MATLAB, Python, FPGA or microcontroller. Applications, advantages, limitations and even possible extensions like Gold codes and hardware realization are also discussed in the paper. The focus is not on an engineering design that was never actually carried out but rather one that can be reproduced.

Keywords— Autocorrelation, feedback polynomial, finite field, LFSR, maximal-length sequence, pseudo-noise, spread spectrum.

I. INTRODUCTION

Digital communication systems often need to use binary sequences which seem noise like and can be Noise-Like Sequences (NLS) trotted.

Produced and multiplied under controlled conditions. Such sequences are called pseudo-noise sequences. A PN sequence is a deterministic sequence, meaning that, once its feedback structure and its initial state are learned, its bits are predetermined for the future. The randomness-like characteristics and the deterministic reproduction of PN sequences make them applicable in communication, synchronization and testing.

The linear feedback shift register (LFSR) is among the most basic, and most studied, PN generators. LFSR consists of a series of binary storage elements and a XOR feedback network. The value in the register is moved at every clock edge and a new bit is generated from selected stages at that time. The generator can visit all non-zero states of the generator before it repeats itself with a proper feedback polynomial.

The PN generator is a helpful example for undergraduate engineering students as it involves ideas from digital electronics, Boolean algebra, communication theory and finite-field mathematics. The circuit level description, simulation, and implementation in programmable hardware are examples of applications of the circuit. It thus bridges the gap between theory and real-world digital-system design in a useful way.

The primary aim of this paper is to investigate the PN sequence generator using an LFSR and explain the influence of the parameters on the period and statistical behavior of the sequence. The paper also outlines a procedure that can be followed to implement it, which is easily applicable in practice. verifying the design without presenting unperformed experimental results as measured data.

A. Objective of the study

The objectives of the study are as follows:

The goals are: to learn how an LFSR works; to learn how the feedback polynomial relates to the period of a sequence; to learn how to make a representative four stage PN generator; to learn the major statistical characteristics of a maximal-length sequence; to compare generators of different length; and to understand a simulation and hardware verification technique.

B. Scope

In the study we will concentrate on binary PN sequences generated by linear feedback shift registers. It includes maximum length sequences and some of their properties. Only as possible extensions are discussed more advanced families like Gold and Kasami sequences. The cryptographic random number generator is not a part of this discussion and is not a secure cryptographic primitive alone since an LFSR is deterministic.

II. BACKGROUND AND LITERATURE REVIEW

The study of shift-register sequences is closely related to the study of polynomial arithmetic over finite fields. Shift-register techniques are an effective way to create long binary sequences with a limited number of registers.

The classical papers on shift-register sequences introduced the conditions under which an LFSR can be made to generate a maximal-length sequence and outlined some important statistical characteristics of these sequences.

Golomb's article on shift-register sequences is a basic reference for the study of m-sequences, their correlation and run properties. Some communication texts, e.g., PN sequences described in Sklar, Proakis and Salehi, and Haykin texts are related to the spreading, synchronization and digital communication systems. The following sources explain the advantages of using a sequence that has low off-peak correlation when a receiver needs to detect a known sequence.

Literature indicates that from an engineering design point of view, a PN generator cannot solely be evaluated based on the generation of changing bits. The requirements of a good design are: a recurrence relation that is clearly documented, a nonzero initial term, a known period and a reproducible output. Theoretical properties can then be checked against a simulation prior to hardware construction.

Relationship Between PN Sequences and Random Noise

While the PN sequence is not strictly random, that is, it is not random in the mathematical sense, it is certainly not random in the sense of being arbitrary. If the generator's structure and state is known, the sequence can be determined. It is called pseudo-noise because the sequence may have noise-like statistical properties for a limited time of observation. It is important to note that this is different from an engineering report, where a deterministic LFSR might be called a random noise source.

Written Work is a compulsory component of the course.

There is a need for a reference pattern that can be created at both the transmitter and the receiver. The LFSR can be conveniently used for this purpose due to the fact that the same seed can be used to produce the same sequence. The receiver is able to correlate its locally generated sequence with the received signal and based on the correlation behavior, align the received signal and/or detect a known spreading pattern.

III. MATHEMATICAL FOUNDATION

A. Binary Arithmetic and GF(2)

It is natural to represent the LFSR feedback in arithmetic mod 2. The elements of the field GF(2) are 0 and 1. Addition is equivalent to the XOR operation: $0 \text{ XOR } 0 = 0$, $0 \text{ XOR } 1 = 1$, $1 \text{ XOR } 0 = 1$ and $1 \text{ XOR } 1 = 0$. No carry in this addition. This is a nice property of GF(2) for digital hardware.

A feedback polynomial is of the form $C(x) = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0$ where each coefficient c_i is either zero or one. The nonzero coefficients give the stages involved in the recurrence. LFSRs use various conventions in the mapping of polynomial coefficients to the physical taps, and the direction of the shifts as well as the arrangement of the taps should be explicitly indicated in a design report.

B. Primitive Polynomials

Primitive means that a polynomial over GF(2) is used to create the longest possible period in the construction of the LFSR. The period of an n-stage LFSR can be as large as $2^n - 1$ if the sequence is started in a non-zero state and if the primitive polynomial used is appropriate. However, a polynomial which is not primitive may be able to generate a shorter cycle despite the n stages of the register.

C. Maximum Period

An n-bit register can have 2^n different states. One state is the state where all the state variables are 0. If the length of the LFSR is maximal, the other $2^n - 1$ states will be in a single cycle. So that the maximum period increases exponentially as n increases. For n=4, the maximum period is 15; for n=8 it is 255; and for n=16 it is 65,535.

D. Seed Selection

The first state is called "seed". For the simple conventional XOR-feedback maximal length LFSR, a non-zero value is needed as a starting seed. Usually, different non-zero seeds produce cyclic shifts between different instances of the same sequence of maximal length when the feedback polynomial remains fixed. Therefore, the seed should be documented if a bit-by-bit reproduction is necessary.

IV. LFSR ARCHITECTURE AND WORKING PRINCIPLE

A. Basic Architecture

The basic PN generator consists of a clock, an n-bit shift register, an n-bit feedback network of XOR gates, an initialization or reset circuit and an output. The speed of new sequence bits is set by the clock. The state is stored in the register, and the next state is calculated in the XOR network.

B. Shift Operation

In a clock event, the contents of the register will shift one bit in the specified direction. Feedback bit is the sum of the taps and is placed in the hole. Depending on the design convention, the output can be either before or after the shift.

This can lead to different-looking sequences when two implementations use the same polynomial name but are mathematically valid.

C. State-Machine Interpretation

LFSRs can also be viewed as finite-state machines. The registers contain all the non-zero values, each of which is a state; the function which feeds back to the registers is the next state. A maximal-length design forms one long cycle, with all the non-zero states within it. This perspective is useful for verifying the design in HDL or software as it is a sequence of state transitions.

D. Block Diagram Description

The practical block diagram will take the form of clock source → shift-register stages → output; selected register taps → XOR feedback network → input of the first register stage. The register is set up by a separate seed-loading block. This block diagram should be followed by a schematic or RTL diagram for a project report, showing *the specific taps*.

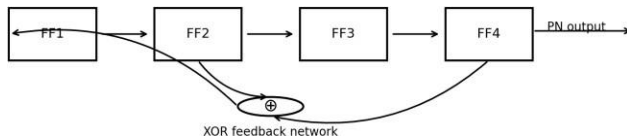


Fig. 1. Basic architecture of an LFSR-based PN sequence generator.

V. DESIGN METHODOLOGY

A. Requirement Identification

The initial design decision is the amount of time that is required for the sequence. If the number of bits L required for the application before the repetition is reached is $\geq L$, then the smallest integer n such that $2^n - 1 \geq L$ is chosen. This gives a systematic way of picking the length of the register, rather than an arbitrary number of stages.

B. Polynomial Selection

A primitive polynomial f of degree n is chosen after picking n . The polynomial should be compatible with the desired LFSRs architecture. The chosen polynomial should be written in the report and the positions of the taps in the implementation should be included.

C. Seed and Output Definition

A seed is chosen and noted that is not zero. The output stage is also indicated. For instance, the most important register bit could be the output and a right shift operation could be used. If they are written down they can be reproduced by another student or engineer.

D. Verification Flow

The suggested verification method is: initialize the register, produce one bit per clock, save register state visited, detect the first repeat of the initial state, count the period, compute some statistics about the sequence. When the period is shorter than anticipated, the most common design problems are using a wrong polynomial, tap convention, shift direction or initialization.

E. Software-to-Hardware Flow

It is necessary to create a software reference model first. Once the state transitions are confirmed, the same recurrence can be programmed with Verilog or VHDL code for an FPGA, or in the bitwise operations of a microcontroller. The first few dozen bits of output from the software and hardware implementations can be used to detect mismatches between the two.

VI. FOUR-STAGE LFSR WORKED EXAMPLE

A. Example Parameters

If a four-stage LFSR is based on the primitive polynomial $x^4 + x + 1$ and starts with a non-zero initial state, then under one well-defined shift convention the feedback is obtained from the appropriate taps. Since different textbooks and hardware descriptions use different tap conventions, it is necessary to specify the exact recurrence relation before asserting that a particular sequence has been generated. The significant theoretical finding is that a properly implemented primitive four-stage LFSR has a period of 15.

B. State Enumeration

A register consisting of four bits has 16 possible states, ranging from 0000 to 1111; in the case of the conventional XOR-feedback maximal-length sequence the state 0000 is excluded and the other 15 states can be arranged in a single cycle if the chosen polynomial is primitive and the implementation is correct.

The student could carry out a practical laboratory experiment by making a table having columns for clock number, current state, feedback bit, next state and output bit. Beginning at the selected seed, the table should be extended until the original seed reappears. In this way, direct evidence of the period of the sequence is obtained and any errors in the taps are easy to identify.

C. Example Verification

The first criterion for verification is that the original state must not reappear before the 15th transition; the second is that each of the 15 non-zero states should occur exactly once in the cycle; the third is that the all-zero state must never be reached. These simple tests offer a solid validation of the LFSR logic at an undergraduate level.

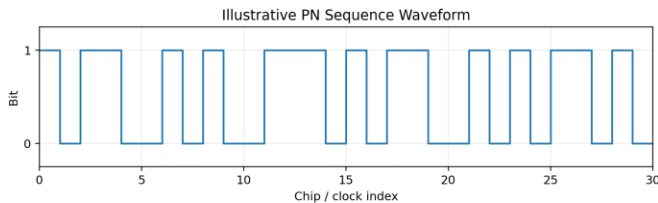


Fig. 2. Illustrative PN sequence waveform.

VII. PN SEQUENCE PROPERTIES

A. Balance Property

In one complete cycle a maximum-length binary sequence has approximately the same number of ones and zeros; in the case of an n -stage m-sequence with a period of $2^n - 1$ there are 2^{n-1} ones and $2^{n-1} - 1$ zeros. This one-bit discrepancy is a direct result of the structure of the maximal-length sequence.

B. Run Property

A run consists of a number of consecutive identical bits; in an m-sequence short runs are more common than long runs and this arrangement of run lengths is what gives the sequence its noise-like appearance. The distribution can be checked by means of a simulation, which involves scanning the stream of generated bits and counting the runs of length one, two, three and so on.

C. Autocorrelation

Autocorrelation involves comparing a sequence with copies of itself which have been shifted. With an m-sequence expressed in bipolar form using the values +1 and -1, the periodic autocorrelation has a large value when the shift is zero and a small constant value for all non-zero shifts. It is therefore useful for synchronization since correct alignment results in a strong correlation peak.

D. Cross-Correlation

It is important to consider cross-correlation when more than one PN sequence is used in a system, since cross-correlation gives a measure of the similarity between two different sequences as one is shifted with respect to the other.

When it is necessary to distinguish between signals or codes, a low value of cross-correlation is desired. In some cases simple m-sequences are not adequate to meet all the requirements of multiple users, which is why code families such as Gold codes have been developed.

E. Periodicity

The sequence repeats after a definite number of clock cycles. Although periodicity is useful when a receiver has to reproduce the sequence, it also implies that the sequence cannot offer true unpredictability. The period has to be long enough for the intended application and must be compatible with the synchronisation strategy.

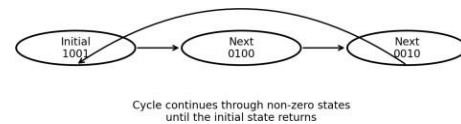


Fig. 3. Illustrative state-transition concept for a four-stage LFSR.

Figure note— The figures in this version are engineering figures meant to explain the generator architecture and the expected sequence behavior. They should not be taken as measured laboratory results. A plot from a simulation or actual hardware can be used to replace these when experimental data are available.

VIII. SIMULATION AND EXPERIMENTAL METHODOLOGY

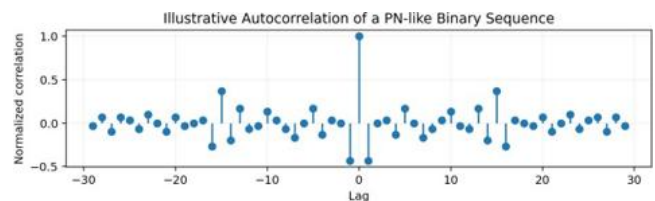


Fig. 4. Illustrative autocorrelation behavior of a PN-like binary sequence.

IX. PERFORMANCE ANALYSIS

A. Effect of Register Length

The register length has the most direct effect on period. Doubling the number of stages does not simply double the period, but rather the maximum period follows $2^n - 1$ and exponentially increases. This is the main reason that LFSRs can create long sequences with relatively little hardware.

B. Hardware Complexity

The storage requirement is roughly n flip-flops for an n -stage LFSR. The feedback network requires XOR operations corresponding to the selected taps.

A sparse primitive polynomial can reduce the number of XOR inputs and hence the complexity of the hardware, although the best option will also have to do with the implementation platform.

C. Speed Considerations

LFSRs are well suited to high-speed digital systems since each clock cycle only has to do a few simple operations. In FPGA implementations, the critical path will be affected by the depth of the XOR network. Large fan-in XOR structures may be balanced or pipelined if a high clock frequency is required..

D. Power and Area Considerations

The small state storage and simple feedback logic will generally make an LFSR compact. The dynamic power depends on the clock frequency, switching activity, and the implementation technology. In a low-power embedded system, the clock rate can be chosen based on the required sequence bit rate.

E. Comparison of Register Lengths

TABLE I
THEORETICAL LFSR PERIOD VERSUS REGISTER LENGTH

Stages n	Maximum Period $2^n - 1$	Storage Elements
3	7	3
4	15	4
5	31	5
6	63	6
8	255	8
10	1,023	10
12	4,095	12
16	65,535	16

X. APPLICATIONS

A. Direct-Sequence Spread Spectrum

In direct-sequence spread-spectrum communication, a high-rate PN sequence is combined with the data sequence to spread the transmitted signal over a larger bandwidth. A receiver that knows the spreading sequence can do a matching despreading operation. The processing gain and synchronization behavior will depend on system parameters beyond the PN generator itself.

B. Synchronization

PN sequences can give recognizable correlation peaks that can help find timing alignment. The receiver can compare the received signal with a locally generated reference and find the shift producing the highest correlation.

C. Ranging and Identification

Known sequences can be sent and correlated at a receiver to find timing relationships. Similar principles can be used in ranging and identification systems, although practical systems may use specialized code families and extra signal processing.

D. Digital Circuit Testing

An LFSR can create pseudorandom test patterns for digital circuits. Instead of storing a big list of test vectors, a small compact LFSR can create a long deterministic stream. This is useful in built-in self-test concepts and laboratory verification.

E. Scrambling

Scrambling changes the appearance of a data stream to avoid long runs or undesirable patterns. An LFSR can be used when the scrambling polynomial and synchronization method are defined by the communication standard. The exact requirements will depend on the application.

XI. ADVANTAGES, LIMITATIONS AND PRACTICAL ISSUES

A. Advantages

The major advantage of an LFSR PN generator are simple hardware, low storage requirement, predictable timing, reproducibility and long period relative to the number of store elements. The circuit is also easy to scale by changing the register length and feedback polynomial.

B. Limitations

The sequence is deterministic and therefore does not provide cryptographic security. A wrong tap configuration can produce a shorter period. The all-zero state can lock a conventional XOR-feedback generator. Different bit-order conventions can also cause confusion when comparing two implementations.

C. Design Precautions

The polynomial, seed, shift direction and output stage should be documented in every implementation. The design should be simulated before hardware testing. Reset behavior should be verified so that the register is never unintentionally left at zero. For communication applications, synchronization between transmitter and receiver must also be considered

XII. COMPARATIVE DISCUSSION

An LFSR-based generator can be compared with a stored lookup-table sequence and with a true random source.

A lookup table offers exact repeatability but needs memory equal to length of stored sequence. An LFSR saves only the current state and generates successive bits by feedback logic. A true random source can have non-deterministic entropy, but it can need a physical noise source and conditioning circuitry. The proper generator is based on the engineering requirement. When repeatability, compact implementation and predictable timing are important, an LFSR is attractive. When security or non-deterministic entropy is needed, another architecture is required. This distinction should be included in an undergraduate project report to demonstrate understanding of application's requirements instead of treating all random-looking sequences as equivalent.

XI. FUTURE SCOPE

The current study can be extended in various ways. First, using MATLAB or Python, one can generate and plot entire PN sequence, its histogram, run distribution and autocorrelation. Second, one can implement the design in Verilog or VHDL and synthesize on an FPGA. Third, different primitive polynomials can be compared for XOR count and maximum clock frequency. Fourth, multiple PN sequences can be studied for cross-correlation.

Another extension is to study of Gold codes and other code families useful for cases when multiple sequences with controlled cross correlation are needed. Another possible extension is integration of the PN generator to a complete spread-spectrum transmitter and receiver, allowing the synchronization and bit-error-rate performance to be studied under noise and interference.

For an undergraduate final-year project, these extensions can transform the theoretical paper into an experimental study by adding measured waveforms, simulation screenshots, hardware resource reports and correlation plots.

XII. CONCLUSION

This paper presented an expanded undergraduate study of a pseudo-noise sequence generator based on use of a linear feedback shift register.

The LFSR provides a compact and efficient way of generating deterministic binary sequences having noise-like statistical characteristics. By choosing a suitable primitive feedback polynomial and a non-zero seed, an n-stage generator can have the maximum theoretical period of $2^n - 1$.

The study discussed the mathematical foundation, architecture, design methodology, four-stage example, sequence properties, simulation procedure, performance factors and engineering applications. This work stressed the importance that exact tap conventions and initialization details must be documented because these determine the generated bit stream. The comparison of register lengths shows that LFSRs provide exponential growth in sequence period with linear growth in storage elements.

The work is suitable as an undergraduate engineering foundation and can be enhanced further by MATLAB/Python simulation and FPGA implementation. The experimental results, when actually measured, can be added to the evaluation section instead of changing the underlying theoretical framework.

Acknowledgment

The author would like to thank the faculty of the Department of Electronics and Telecommunication Engineering, MVPS's KBT College of Engineering, Nashik, for their academic guidance and support.

REFERENCES

- [1] S. W. Golomb, Shift Register Sequences, revised ed., Aegean Park Press, 1982.
- [2] R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications, Cambridge University Press, 1994.
- [3] B. Sklar, Digital Communications: Fundamentals and Applications, 2nd ed., Prentice Hall, 2001.
- [4] J. G. Proakis and M. Salehi, Digital Communications, 5th ed., McGraw-Hill, 2008.
- [5] S. Haykin, Communication Systems, 4th ed., John Wiley & Sons, 2001.
- [6] R. A. Rueppel, Analysis and Design of Stream Ciphers, Springer-Verlag, 1986.
- [7] D. E. Knuth, The Art of Computer Programming, Vol. 2: Semi numerical Algorithms, 3rd ed., Addison-Wesley, 1997.
- [8] W. Stallings, Data and Computer Communications, 10th ed., Pearson, 2013.