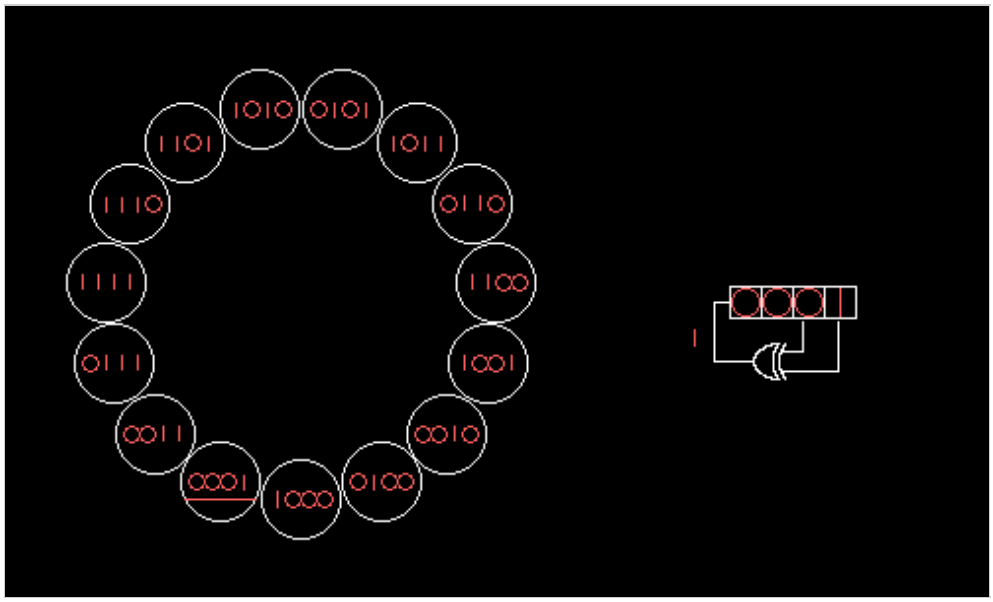
# Linear feedback shift register

From Wikipedia, the free encyclopedia

In computing, a **linear-feedback shift register** (LFSR) is a shift register whose input bit is a linear function of its previous state.




A 4-bit Fibonacci LFSR with its state diagram. The XOR gate provides feedback to the register that shifts bits from left to right. The maximal sequence consists of every possible state except the "0000" state.

The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value.

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle.

Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common.

The mathematics of a cyclic redundancy check, used to provide a quick check against transmission errors, are closely related to those of an LFSR.

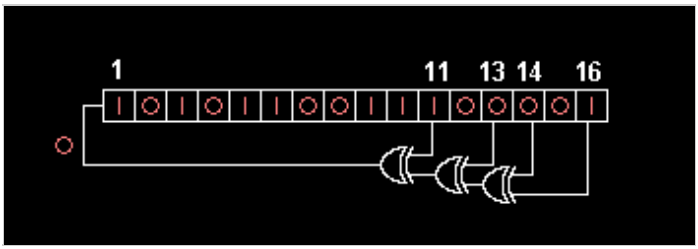
## Contents


- 1 Fibonacci LFSRs
- 2 Galois LFSRs
  - 2.1 Non-binary Galois LFSR
- 3 Some polynomials for maximal LFSRs
- 4 Output-stream properties
- 5 Applications
  - 5.1 Uses as counters
  - 5.2 Uses in cryptography
  - 5.3 Uses in circuit testing
    - 5.3.1 Test-pattern generation
    - 5.3.2 Signature analysis

- 5.4 Uses in digital broadcasting and communications
  - 5.4.1 Scrambling
  - 5.4.2 Other uses
- 6 See also
- 7 References
- 8 External links


# Fibonacci LFSRs

The bit positions that affect the next state are called the taps. In the diagram the taps are [16,14,13,11]. The rightmost bit of the LFSR is called the output bit. The taps are XOR'd sequentially with the output bit and then fed back into the leftmost bit. The sequence of bits in the rightmost position is called the output stream.




A 16-bit Fibonacci LFSR. The feedback tap numbers in white correspond to a primitive polynomial in the table so the register cycles through the maximum number of 65535 states excluding the all-zeroes state. The state shown, 0xACE1 (hexadecimal) will be followed by 0x5670.

- The bits in the LFSR state which influence the input are called *taps* (white in the diagram).
- A maximum-length LFSR produces an m-sequence (i.e. it cycles through all possible $2^n - 1$ states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change.
- As an alternative to the XOR based feedback in an LFSR, one can also use XNOR.[1] This function is an affine map, not strictly a linear map, but it results in an equivalent polynomial counter whose state of this counter is the complement of the state of an LFSR. A state with all ones is illegal when using an XNOR feedback, in the same way as a state with all zeroes is illegal when using XOR. This state is considered illegal because the counter would remain "locked-up" in this state.

The sequence of numbers generated by an LFSR or its XNOR counterpart can be considered a binary numeral system just as valid as Gray code or the natural binary code.

The arrangement of taps for feedback in an LFSR can be expressed in finite field arithmetic as a polynomial mod 2. This means that the coefficients of the polynomial must be 1's or 0's. This is called the feedback polynomial or reciprocal characteristic polynomial. For example, if the taps are at the 16th, 14th, 13th and 11th bits (as shown), the feedback polynomial is

$$x^{16} + x^{14} + x^{13} + x^{11} + 1.$$

The 'one' in the polynomial does not correspond to a tap – it corresponds to the input to the first bit (i.e. $x^0$, which is equivalent to 1). The powers of the terms represent the tapped bits, counting from the left. The first and last bits are always connected as an input and output tap respectively.

The LFSR is maximal-length if and only if the corresponding feedback polynomial is primitive. This means that the following conditions are necessary (but not sufficient):

- The number of taps should be even.
- The set of taps – taken all together, *not* pairwise (i.e. as pairs of elements) – must be relatively prime. In other words, there must be no divisor other than 1 common to all taps.

Tables of primitive polynomials from which maximum-length LFSRs can be constructed are given below and in the references.

There can be more than one maximum-length tap sequence for a given LFSR length. Also, once one maximum-length tap sequence has been found, another automatically follows. If the tap sequence, in an $n$-bit LFSR, is $[n, A, B, C, 0]$, where the 0 corresponds to the $x^0 = 1$ term, then the corresponding 'mirror' sequence is $[n, n - C, n - B, n - A, 0]$. So the tap sequence [32, 7, 3, 2, 0] has as its counterpart [32, 30, 29, 25, 0]. Both give a maximum-length sequence.

Some example C code is below:

```c
# include <stdint.h>
int main(void)
{
    uint16_t start_state = 0xACE1u;  /* Any nonzero start state will work. */
    uint16_t lfsr = start_state;
    unsigned bit;
    unsigned period = 0;

    do
    {
        /* taps: 16 14 13 11; feedback polynomial: x^16 + x^14 + x^13 + x^11 + 1 */
        bit  = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ) & 1;
        lfsr =  (lfsr >> 1) | (bit << 15);
        ++period;
    } while (lfsr != start_state);

    return 0;
}
```

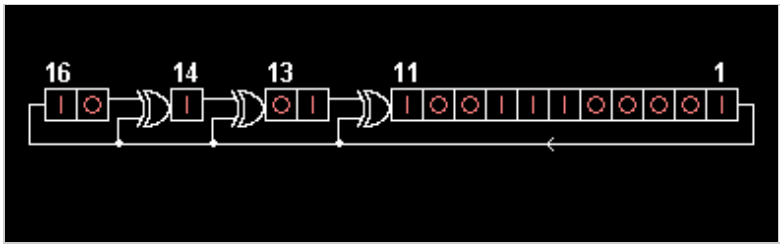
This LFSR configuration is also known as **standard**, **many-to-one** or **external XOR gates**. The alternative Galois configuration is described in the next section.

# Galois LFSRs

Named after the French mathematician Évariste Galois, an LFSR in Galois configuration, which is also known as **modular**, **internal XORs** as well as **one-to-many LFSR**, is an alternate structure that can generate the same output stream as a conventional LFSR (but offset in time).[2] In the Galois configuration, when the system is clocked, bits that are not taps are shifted one position to the right unchanged. The taps, on the other hand, are XOR'd with the output bit before they are stored in the next position. The new output bit is the next input bit. The effect of this is that when the output bit is zero all the bits in the register shift to the right unchanged, and the input bit becomes zero. When the output bit is one, the bits in the tap positions all flip (if they are 0, they become 1, and if they are 1, they become 0), and then the entire register is shifted to the right and the input bit becomes 1.

To generate the same output stream, the order of the taps is the *counterpart* (see above) of the order for the conventional LFSR, otherwise the stream will be in reverse. Note that the internal state of the LFSR is not necessarily the same. The Galois register shown has the same output stream as the Fibonacci register in the first section. A time offset exists between the streams, so a different startpoint will be needed to get the same output each cycle.

- Galois LFSRs do not concatenate every tap to produce the new input (the XOR'ing is done within the LFSR and no XOR gates are run in serial, therefore the propagation times are reduced to that of one XOR rather than a whole chain), thus it is possible for each tap to be computed in parallel, increasing the speed of execution.
- In a software implementation of an LFSR, the Galois form is more efficient as the XOR operations can be implemented a word at a time: only the output bit must be examined individually.



A 16-bit Galois LFSR. The register numbers in white correspond to the same primitive polynomial as the Fibonacci example but are counted in reverse to the shifting direction. This register also cycles through the maximal number of 65535 states excluding the all-zeroes state. The state ACE1 hex shown will be followed by E270 hex.

Below is a C code example for the 16 bit maximal period Galois LFSR example in the figure:

```c
# include <stdint.h>
int main(void)
{
    uint16_t start_state = 0xACE1u;  /* Any nonzero start state will work. */
    uint16_t lfsr = start_state;
    unsigned period = 0;

    do
    {
        unsigned lsb = lfsr & 1;  /* Get LSB (i.e., the output bit). */
        lfsr >>= 1;               /* Shift register */
        if (lsb == 1)             /* Only apply toggle mask if output bit is 1. */
            lfsr ^= 0xB400u;      /* Apply toggle mask, value has 1 at bits corresponding
                                   * to taps, 0 elsewhere. */
        ++period;
    } while (lfsr != start_state);

    return 0;
}
```

### Non-binary Galois LFSR

Binary Galois LFSRs like the ones shown above can be generalized to any $q$-ary alphabet $\{0, 1, ..., q-1\}$ (e.g., for binary, $q$ is equal to two, and the alphabet is simply $\{0, 1\}$). In this case, the exclusive-or component is generalized to addition modulo-$q$ (note that XOR is addition modulo 2), and the feedback bit (output bit) is multiplied (modulo-$q$) by a $q$-ary value which is constant for each specific tap point. Note that this is also a generalization of the binary case, where the feedback is multiplied by either 0 (no feedback, i.e., no tap) or 1 (feedback is present). Given an appropriate tap configuration, such LFSRs can be used to generate Galois fields for arbitrary prime values of $q$.

## Some polynomials for maximal LFSRs

The following table lists maximal-length polynomials for shift-register lengths up to 19. Note that more than one maximal-length polynomial may exist for any given shift-register length. A list of alternative maximal-length polynomials for shift-register lengths 4-32 (beyond which it becomes unfeasible to store or transfer them) can be found here: http://www.ece.cmu.edu/~koopman/lfsr/index.html

| Bits | Feedback polynomial | Period |
|---|---|---|
| ***n*** | | $2^n - 1$ |
| **2** | $x^2 + x + 1$ | 3 |
| **3** | $x^3 + x^2 + 1$ | 7 |
| **4** | $x^4 + x^3 + 1$ | 15 |
| **5** | $x^5 + x^3 + 1$ | 31 |
| **6** | $x^6 + x^5 + 1$ | 63 |
| **7** | $x^7 + x^6 + 1$ | 127 |
| **8** | $x^8 + x^6 + x^5 + x^4 + 1$ | 255 |
| **9** | $x^9 + x^5 + 1$ | 511 |
| **10** | $x^{10} + x^7 + 1$ | 1023 |
| **11** | $x^{11} + x^9 + 1$ | 2047 |
| **12** | $x^{12} + x^{11} + x^{10} + x^4 + 1$ | 4095 |
| **13** | $x^{13} + x^{12} + x^{11} + x^8 + 1$ | 8191 |
| **14** | $x^{14} + x^{13} + x^{12} + x^2 + 1$ | 16383 |
| **15** | $x^{15} + x^{14} + 1$ | 32767 |
| **16** | $x^{16} + x^{14} + x^{13} + x^{11} + 1$ | 65535 |
| **17** | $x^{17} + x^{14} + 1$ | 131071 |
| **18** | $x^{18} + x^{11} + 1$ | 262143 |
| **19** | $x^{19} + x^{18} + x^{17} + x^{14} + 1$ | 524287 |
| **20-168** | [2] (http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf) | |
| **2-786,1024,2048,4096** | [3] (http://www.eej.ulst.ac.uk/~ian/modules/EEE515/files/old_files/lfsr/lfsr_table.pdf) | |

# Output-stream properties

- Ones and zeroes occur in 'runs'. The output stream 1110010, for example consists of four runs of lengths 3,2,1,1, in order. In one period of a maximal LFSR, $2^{n-1}$ runs occur (for example, a six bit LFSR will have 32 runs). Exactly half of these runs will be one bit long, a quarter will be two bits long, up to a single run of zeroes $n - 1$ bits long, and a single run of ones $n$ bits long. This distribution almost equals the statistical expectation value for a truly random sequence. However, the probability of finding exactly this distribution in a sample of a truly random sequence is rather low.
- LFSR output streams are deterministic. If you know the present state as well as the positions of the XOR gates in the LFSR, you can predict the next state.[3] This is not possible with truly random events. With minimal-length LFSRs, it is much easier to compute the next state, as there are only an easily limited number of them for each length.
- The output stream is reversible; an LFSR with mirrored taps will cycle through the output sequence in reverse order.

# Applications

LFSRs can be implemented in hardware, and this makes them useful in applications that require very fast generation of a pseudo-random sequence, such as direct-sequence spread spectrum radio. LFSRs have also been used for generating an approximation of white noise in various programmable sound generators.

## Uses as counters

The repeating sequence of states of an LFSR allows it to be used as a clock divider, or as a counter when a non-binary sequence is acceptable as is often the case where computer index or framing locations need to be machine-readable.[3] LFSR counters have simpler feedback logic than natural binary counters or Gray code counters, and therefore can operate at higher clock rates. However it is necessary to ensure that the LFSR never enters an all-zeros state, for example by presetting it at start-up to any other state in the sequence. The table of primitive polynomials shows how LFSRs can be arranged in Fibonacci or Galois form to give maximal periods. One can obtain any other period by adding to an LFSR that has a longer period some logic that shortens the sequence by skipping some states.

## Uses in cryptography

LFSRs have long been used as pseudo-random number generators for use in stream ciphers (especially in military cryptography), due to the ease of construction from simple electromechanical or electronic circuits, long periods, and very uniformly distributed output streams. However, an LFSR is a linear system, leading to fairly easy cryptanalysis. For example, given a stretch of known plaintext and corresponding ciphertext, an attacker can intercept and recover a stretch of LFSR output stream used in the system described, and from that stretch of the output stream can construct an LFSR of minimal size that simulates the intended receiver by using the Berlekamp-Massey algorithm. This LFSR can then be fed the intercepted stretch of output stream to recover the remaining plaintext.

Three general methods are employed to reduce this problem in LFSR-based stream ciphers:

- Non-linear combination of several bits from the LFSR state;
- Non-linear combination of the output bits of two or more LFSRs (see also: shrinking generator); or using Evolutionary algorithm to introduce non-linearity.[4]
- Irregular clocking of the LFSR, as in the alternating step generator.

Important LFSR-based stream ciphers include A5/1 and A5/2, used in GSM cell phones, E0, used in Bluetooth, and the shrinking generator. The A5/2 cipher has been broken and both A5/1 and E0 have serious weaknesses.[5][6]

The linear feedback shift register has a strong relationship to linear congruential generators.[7]

## Uses in circuit testing

LFSRs are used in circuit testing, for test-pattern generation (for exhaustive testing, pseudo-random testing or pseudo-exhaustive testing) and for signature analysis.

### Test-pattern generation

Complete LFSR are commonly used as pattern generators for exhaustive testing, since they cover all possible inputs for an n input circuit. Maximum length LFSRs and weighted LFSRs are widely used as pseudo-random test pattern generators for pseudo-random test applications.

### Signature analysis

In built-in self-test (BIST) techniques, storing all the circuit outputs on chip is not possible, but the circuit output can be compressed to form a signature which later will be compared to the golden signature (of the good circuit) to detect faults. Since this compression is lossy, there is always a probability that a faulty output also generates the same signature as the golden signature and the faults can not be detected. This condition is called error masking or aliasing. This is accomplished by using a multiple-input signature register (MISR or MSR) which is a type of LFSR. A standard LFSR has a single XOR or XNOR gate where the input of the gate is connected to several "taps" and the output is connected to the input of the first flip-flop. A MISR has the same structure, however, the input to every flip-flop is fed through an XOR/XNOR gate. For example, a four bit MISR has a four-bit parallel output and a four-bit parallel input. The input of the first flip-flop is XOR/XNORd with parallel input bit zero and the "taps." Every other flip-flop input is XOR/XNORd with the preceding flip-flop output and the corresponding parallel input bit. Consequently, the next state of the MISR is dependent on the last several states opposed to just the current state. Therefore, a MISR will always generate the same golden signature given that input sequence is the same every time.

## Uses in digital broadcasting and communications

### Scrambling

To prevent short repeating sequences (e.g., runs of 0's or 1's) from forming spectral lines that may complicate symbol tracking at the receiver or interfere with other transmissions, linear feedback registers are often used to "randomize" the transmitted bitstream. This randomization is removed at the receiver after demodulation. When the LFSR runs at the same bit rate as the transmitted symbol stream, this technique is referred to as scrambling. When the LFSR runs considerably faster than the symbol stream, expanding the bandwidth of the transmitted signal, this is direct-sequence spread spectrum.

Neither scheme should be confused with encryption or encipherment; scrambling and spreading with LFSRs do *not* protect the information from eavesdropping. They are instead used to produce equivalent streams that possess convenient engineering properties to allow for robust and efficient modulation and demodulation.

Digital broadcasting systems that use linear feedback registers:

- ATSC Standards (digital TV transmission system – North America)
- DAB (Digital Audio Broadcasting system – for radio)
- DVB-T (digital TV transmission system – Europe, Australia, parts of Asia)
- NICAM (digital audio system for television)

Other digital communications systems using LFSRs:

- INTELSAT business service (IBS)
- Intermediate data rate (IDR)
- SDI (Serial Digital Interface transmission)
- Data transfer over PSTN (according to the ITU-T V-series recommendations)
- CDMA (Code Division Multiple Access) cellular telephony
- 100BASE-T2 "fast" Ethernet scrambles bits using an LFSR
- 1000BASE-T Ethernet, the most common form of Gigabit Ethernet, scrambles bits using an LFSR
- PCI Express 3.0
- SATA[8]

- Serial attached SCSI (SAS/SPL)
- USB 3.0
- IEEE 802.11a scrambles bits using an LFSR
- Bluetooth Low Energy Link Layer is making use of LFSR (referred to as whitening)

**Other uses**

The German time signal DCF77, in addition to amplitude keying, employs phase-shift keying driven by a 9-stage LFSR to increase the accuracy of received time and the robustness of the data stream in the presence of noise.[9]

The Global Positioning System uses an LFSR to rapidly transmit a sequence that indicates high-precision relative time offsets.

LFSRs are also used in Communications System Jamming systems in which they are used to generate pseudo random noise to raise the noise floor of a target communication system.

# See also

- Pinwheel
- Mersenne twister
- Maximum length sequence
- Analog feedback shift register
- NLFSR, Non-Linear Feedback Shift Register

# References

1. Linear Feedback Shift Registers in Virtex Devices (http://www.xilinx.com/support/documentation/application_notes/xapp210.pdf)
2. Beker, Henry; Piper, Fred (1982). *Cipher Systems: The Protection of Communications*. Wiley-Interscience. p. 212.
3. http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf
4. A. Poorghanad, A. Sadr, A. Kashanipour" Generating High Quality Pseudo Random Number Using Evolutionary Methods", IEEE Congress on Computational Intelligence and Security, vol. 9, pp. 331-335 , May,2008 [1] (http://www.computer.org/csdl/proceedings/cis/2008/3508/01/3508a331.pdf)
5. Barkam, Elad; Biham, Eli; Keller, Nathan (2008), "Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication" (http://cryptome.org/gsm-crack-bbk.pdf), *Journal of Cryptology* **21** (3): 392–429, doi:10.1007/s00145-007-9001-y (https://dx.doi.org/10.1007%2Fs00145-007-9001-y)
6. Lu, Yi; Willi Meier; Serge Vaudenay (2005). "The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption" (http://www.terminodes.org/micsPublicationsDetail.php?pubno=1216). *Crypto 2005* (Santa Barbara, California, USA) **3621**: 97–117. doi:10.1007/11535218_7 (https://dx.doi.org/10.1007%2F11535218_7).
7. RFC 4086 section 6.1.3 "Traditional Pseudo-random Sequences"
8. Section 9.5 of the SATA Specification, revision 2.6
9. Hetzel, P. (16 March 1988). *Time dissemination via the LF transmitter DCF77 using a pseudo-random phase-shift keying of the carrier* (https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/pdf/5_1988_Hetzel_-_Proc_EFTF_88.pdf). 2nd European Frequency and Time Forum. Neuchâtel. pp. 351–364. Retrieved 11 October 2011.

# External links

- LFSR Reference (http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm ) LFSR theory and implementation, maximal length sequences, and comprehensive feedback tables for lengths from 7 to 16,777,215 (3 to 24 stages), and partial tables for lengths up to 4,294,967,295 (25 to 32 stages).
- International Telecommunications Union Recommendation O.151 (http://www.itu.int/rec/T-REC-O.151-199210-I/en) (August 1992)
- Maximal Length LFSR table (http://spreadsheets.google.com/ccc?key=0AvYtZsho-JTldFRYZnJLRFFaSWtUcVNXc1Y3M2VWd1E&hl=en) with length from 2 to 67.
- Pseudo-Random Number Generation Routine (http://www.maxim-ic.com/appnotes.cfm?appnote_number=1743&CMP=WP-9)
- http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/1999f/Drivers_Ed/lfsr.html
- http://www.quadibloc.com/crypto/co040801.htm
- Simple explanation of LFSRs for Engineers (http://www.yikes.com/~ptolemy/lfsr_web/index.htm)
- Feedback terms (http://www.ece.cmu.edu/~koopman/lfsr/index.html)
- General LFSR Theory (http://homepage.mac.com/afj/lfsr.html)
- An implementation of LFSR in VHDL. (http://opencores.org/project,lfsr_randgen)
- Simple VHDL coding for Galois and Fibonacci LFSR. (http://emmanuel.pouly.free.fr)



Categories: Binary arithmetic | Digital registers | Cryptographic algorithms | Pseudorandom number generators

---