Linear Feedback Shift Registers and Complexity A survey

Michele Elia (Politecnico di Torino)

Bunny TN 3

Trento, 12 marzo 2012

Outline

Outline

1. Binary sequences and Complexity

Outline

1. Binary sequences and Complexity
2. Finite State Machines and LFSR

Outline

1. Binary sequences and Complexity
2. Finite State Machines and LFSR
3. HW/SW Complexity

Outline

1. Binary sequences and Complexity
2. Finite State Machines and LFSR
3. HW/SW Complexity
4. LFSR structures

Outline

1. Binary sequences and Complexity
2. Finite State Machines and LFSR
3. HW/SW Complexity
4. LFSR structures
5. LFSR ?

Outline

Binary sequences

$$\ldots, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \ldots$$

$$\ldots, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, \ldots$$

$$\ldots, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \ldots$$

$$\ldots, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, \ldots$$

Binary sequences

$$\ldots, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \ldots$$

$$\ldots, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, \ldots$$

$$\ldots, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \ldots$$

$$\ldots, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, \ldots$$

- **Cryptography**

Binary sequences

$$\ldots, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \ldots$$

$$\ldots, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, \ldots$$

$$\ldots, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \ldots$$

$$\ldots, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, \ldots$$

- **Cryptography**
- **Testing of digital devices**

Binary sequences

$$\ldots, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \ldots$$

$$\ldots, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, \ldots$$

$$\ldots, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \ldots$$

$$\ldots, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, \ldots$$

- **Cryptography**
- **Testing of digital devices**
- **Spread spectrum techniques**

Binary sequences

$$\ldots, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \ldots$$

$$\ldots, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, \ldots$$

$$\ldots, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \ldots$$

$$\ldots, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, \ldots$$

- **Cryptography**
- **Testing of digital devices**
- **Spread spectrum techniques**
- **Navigation** and localization systems

Binary sequences

$$\ldots, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, \ldots$$

$$\ldots, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, \ldots$$

$$\ldots, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \ldots$$

$$\ldots, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, \ldots$$

- **Cryptography**
- **Testing of digital devices**
- **Spread spectrum techniques**
- **Navigation** and localization systems
- **Simulation**

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Examples of Tests

- *Average, Mean-square error, χ-square test*

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Examples of Tests

- *Average, Mean-square error, χ-square test*
- Monte Carlo tests

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Examples of Tests

- *Average, Mean-square error, χ-square test*
- Monte Carlo tests
- *Kolmogorov-Smirnov test*

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Examples of Tests

- *Average, Mean-square error, χ-square test*
- Monte Carlo tests
- *Kolmogorov-Smirnov test*
- Runs' test, Auto-correlation function test

Random sequences (*Knuth, The art of Computer Programming, vol.2*)

Definition (D.H. Lehmer (1951))

A random sequence $x_1, x_2, x_3, x_4, \ldots$ is a sequence such that

1. embodies the idea that each term is unpredictable to the uninitiated observer;
2. its digits pass a certain number of tests traditional with statisticians and depending somewhat on the uses to which the sequence is to be put.

Examples of Tests

- *Average, Mean-square error, χ-square test*
- Monte Carlo tests
- *Kolmogorov-Smirnov test*
- Runs' test, Auto-correlation function test
- *Linear Complexity Profile*

Random sequences and Information measures

Random sequences and Information measures

- The maximum amount of information carried by a binary sequence is equal to its length.

Random sequences and Information measures

- The maximum amount of information carried by a binary sequence is equal to its length.
- A genuine random binary sequence of statistically independent and equiprobable symbols cannot be described using an amount of information smaller than its length.

Random sequences and Information measures

- The maximum amount of information carried by a binary sequence is equal to its length.
- A genuine random binary sequence of statistically independent and equiprobable symbols cannot be described using an amount of information smaller than its length.
- The measure of information carried by a sequence can be taken as a measure of its complexity: it follows that *genuine random binary sequences* are sequences of maximum complexity.

Random sequences and Information measures

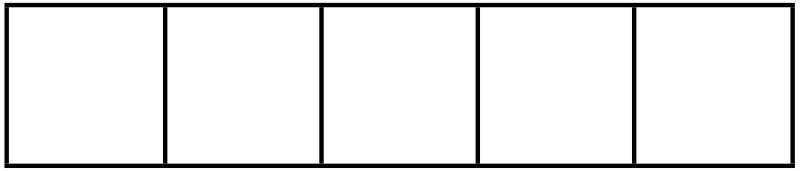
- The maximum amount of information carried by a binary sequence is equal to its length.
- A genuine random binary sequence of statistically independent and equiprobable symbols cannot be described using an amount of information smaller than its length.
- The measure of information carried by a sequence can be taken as a measure of its complexity: it follows that *genuine random binary sequences* are sequences of maximum complexity.

Random sequences and Information measures

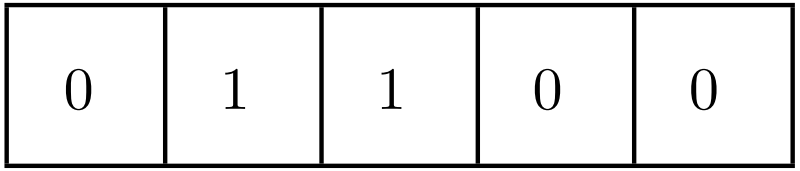
- The maximum amount of information carried by a binary sequence is equal to its length.
- A genuine random binary sequence of statistically independent and equiprobable symbols cannot be described using an amount of information smaller than its length.
- The measure of information carried by a sequence can be taken as a measure of its complexity: it follows that *genuine random binary sequences* are sequences of maximum complexity.

Kolmogorov: the algorithmic complexity description of an object is the length of the shortest binary computer program that describe the object

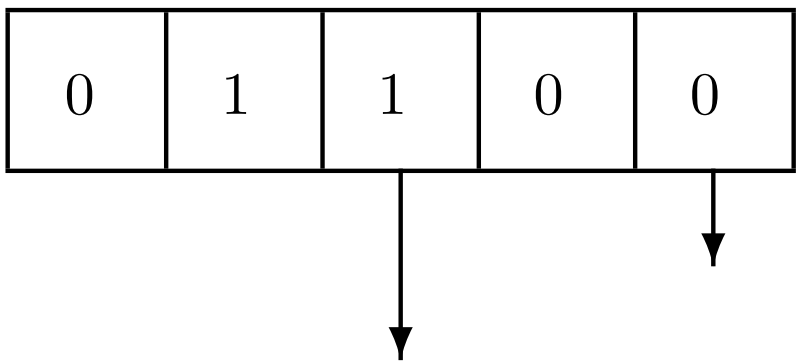
Linear registers



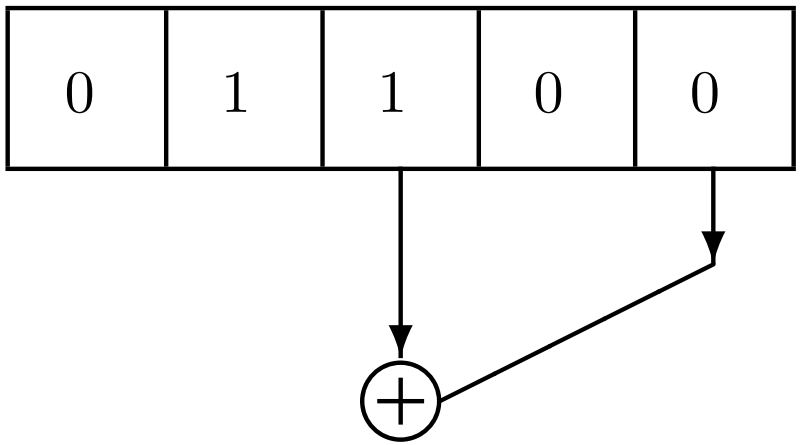
Linear registers



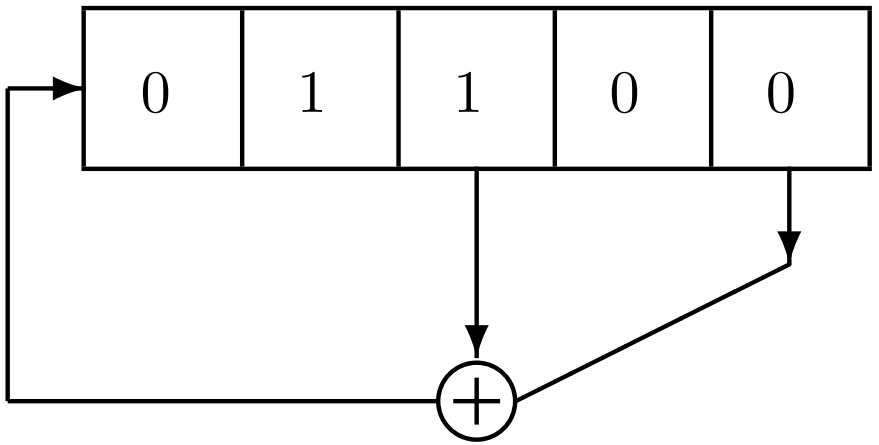
Linear registers



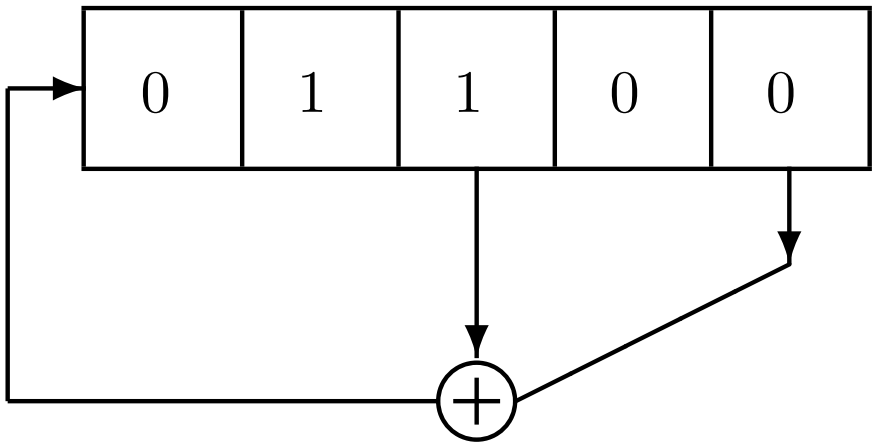
Linear registers



Linear registers



Linear registers



$$g(Z) = 1 + Z^{-2} + Z^{-5}$$

Linear FINITE STATE MACHINE (FSM)

Linear FINITE STATE MACHINE (FSM)

A FSM is a five-tuple $\{\mathcal{A}, \mathcal{S}, stat, out, \mathbf{s}_o\}$ where

Linear FINITE STATE MACHINE (FSM)

A FSM is a five-tuple $\{\mathcal{A}, \mathcal{S}, stat, out, \mathbf{s}_o\}$ where

- $\mathcal{A}$ is the output finite set of symbols (e.g. $\mathbb{F}_2$) .

Linear FINITE STATE MACHINE (FSM)

A FSM is a five-tuple $\{\mathcal{A}, \mathcal{S}, stat, out, \mathbf{s}_o\}$ where

- $\mathcal{A}$ is the output finite set of symbols (e.g. $\mathbb{F}_2$) .
- $\mathcal{S} = \{\mathbf{s}\}$ is the finite set of states (e.g. $\mathbf{s} \in \mathbb{F}_2^m$)

Linear FINITE STATE MACHINE (FSM)

A FSM is a five-tuple $\{\mathcal{A}, \mathcal{S}, stat, out, \mathbf{s}_o\}$ where

- $\mathcal{A}$ is the output finite set of symbols (e.g. $\mathbb{F}_2$) .
- $\mathcal{S} = \{\mathbf{s}\}$ is the finite set of states (e.g. $\mathbf{s} \in \mathbb{F}_2^m$)
- $stat$ is the transition function, that is a mapping from $\mathcal{S}$ into $\mathcal{S}$

$$stat: \quad \mathcal{S} \to \mathcal{S}$$

(e.g. $stat(\mathbf{s}) = \mathfrak{M}\mathbf{s}$ where $\mathfrak{M}$ is an $m \times m$ binary matrix).

Linear FINITE STATE MACHINE (FSM)

A FSM is a five-tuple $\{\mathcal{A}, \mathcal{S}, stat, out, \mathbf{s}_o\}$ where

- $\mathcal{A}$ is the output finite set of symbols (e.g. $\mathbb{F}_2$) .
- $\mathcal{S} = \{\mathbf{s}\}$ is the finite set of states (e.g. $\mathbf{s} \in \mathbb{F}_2^m$)
- $stat$ is the transition function, that is a mapping from $\mathcal{S}$ into $\mathcal{S}$

$$stat: \quad \mathcal{S} \to \mathcal{S}$$

(e.g. $stat(\mathbf{s}) = \mathfrak{M}\mathbf{s}$ where $\mathfrak{M}$ is an $m \times m$ binary matrix).

- out is the output function, that is a mapping from $\mathcal{S}$ into $\mathcal{A}$

$$stat: \quad \mathcal{S} \to \mathcal{A}$$

(e.g. $out(\mathbf{s}) = s_m \in \mathbb{F}_2$, $\mathbf{s}$ is an m-dimensional binary vector)

Linear FINITE STATE MACHINE (FSM)

A FSM is a five-tuple $\{\mathcal{A}, \mathcal{S}, stat, out, \mathbf{s}_o\}$ where

- $\mathcal{A}$ is the output finite set of symbols (e.g. $\mathbb{F}_2$) .
- $\mathcal{S} = \{\mathbf{s}\}$ is the finite set of states (e.g. $\mathbf{s} \in \mathbb{F}_2^m$)
- $stat$ is the transition function, that is a mapping from $\mathcal{S}$ into $\mathcal{S}$

$$stat: \quad \mathcal{S} \to \mathcal{S}$$

(e.g. $stat(\mathbf{s}) = \mathfrak{M}\mathbf{s}$ where $\mathfrak{M}$ is an $m \times m$ binary matrix).

- out is the output function, that is a mapping from $\mathcal{S}$ into $\mathcal{A}$

$$stat: \quad \mathcal{S} \to \mathcal{A}$$

(e.g. $out(\mathbf{s}) = s_m \in \mathbb{F}_2$, $\mathbf{s}$ is an m-dimensional binary vector)

- $\mathbf{s}_o$ is the initial state, i.e. a fixed element from $\mathcal{S}$

Linear Sequences $x(0), x(1), \ldots, x(n), \ldots$

Linear Sequences $x(0), x(1), \ldots, x(n), \ldots$

A linear sequence generated by a FSM may be specified in two ways

Linear Sequences $x(0), x(1), \ldots, x(n), \ldots$

A linear sequence generated by a FSM may be specified in two ways

- Using the matrix description

$$\mathbf{s}(n+1) = \mathfrak{M}\mathbf{s}(n) \quad , \quad x(n+1) = \mathbf{s}_m(n+1)$$

Linear Sequences $x(0), x(1), \ldots, x(n), \ldots$

A linear sequence generated by a FSM may be specified in two ways

- Using the matrix description

$$\mathbf{s}(n+1) = \mathfrak{M}\mathbf{s}(n) \quad , \quad x(n+1) = \mathbf{s}_m(n+1)$$

- Using linear recurrences, i.e. recurrent equations of order m

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \cdots + a_m x(n-m)$$

Generating function

Generating function

The generating function of an infinite sequence is

$$X(Z) = \sum_{n=0}^{\infty} x(n) Z^{-n}$$

Generating function

The generating function of an infinite sequence is

$$X(Z) = \sum_{n=0}^{\infty} x(n) Z^{-n}$$

The generating function of a linear sequence is a rational function, i.e.

$$X(Z) = \frac{b(Z)}{g(Z)} = \frac{b_0 + b_1 Z^{-1} + \cdots + b_{m-1} Z^{-m+1}}{a_m + a_{m-1} Z^{-1} + \cdots + Z^{-m}}$$

where b_i's depend on the initial state (initial conditions).

Generating function

The generating function of an infinite sequence is

$$X(Z) = \sum_{n=0}^{\infty} x(n) Z^{-n}$$

The generating function of a linear sequence is a rational function, i.e.

$$X(Z) = \frac{b(Z)}{g(Z)} = \frac{b_0 + b_1 Z^{-1} + \cdots + b_{m-1} Z^{-m+1}}{a_m + a_{m-1} Z^{-1} + \cdots + Z^{-m}}$$

where b_i's depend on the initial state (initial conditions).

$g(Z)$ is the LFSR polynomial generator, and is also the characteristic polynomial of the transition matrix $\mathfrak{M}$.

Period

Period

The generating function of a periodic sequence of period τ, can be written as

$$X(Z) = \frac{x(0) + x(1)Z^{-1} + \cdots x(\tau-1)Z^{-\tau+1}}{1 - Z^{-\tau}}$$

Period

The generating function of a periodic sequence of period τ, can be written as

$$X(Z) = \frac{x(0) + x(1)Z^{-1} + \cdots x(\tau-1)Z^{-\tau+1}}{1 - Z^{-\tau}}$$

- There is a minimum τ such that $\mathfrak{M}^{\tau} = \mathfrak{I}$ therefore the generated sequences are periodic of period not greater than τ.

  - In general the period depends on the initial state.
  - τ is the minimum integer such that $g(Z)$ divides $1 - Z^{-\tau}$.

Period

The generating function of a periodic sequence of period τ, can be written as

$$X(Z) = \frac{x(0) + x(1)Z^{-1} + \cdots x(\tau-1)Z^{-\tau+1}}{1 - Z^{-\tau}}$$

- There is a minimum τ such that $\mathfrak{M}^{\tau} = \mathfrak{I}$ therefore the generated sequences are periodic of period not greater than τ.

  - In general the period depends on the initial state.
  - τ is the minimum integer such that $g(Z)$ divides $1 - Z^{-\tau}$.
- The maximum value of τ is $2^m - 1$, and is attained by primitive polynomial generators.
  The generated sequences are called m-sequences, in this case the period is independent of the initial state (the all-zeros state is excluded).

LFSR and Cyclic codes

The block of symbols forming a period of an m-sequence generated by a given LFSR can be considered (are) as codewords of a cyclic code which is the dual code

$$(2^m - 1, m, 2^{m-1})$$

of an Hamming code $(2^m - 1, 2^m - 1 - m, 3)$.

LFSR and Cyclic codes

The block of symbols forming a period of an m-sequence generated by a given LFSR can be considered (are) as codewords of a cyclic code which is the dual code

$$(2^m - 1, m, 2^{m-1})$$

of an Hamming code $(2^m - 1, 2^m - 1 - m, 3)$.
Every non-zero code word of a dual Hamming code has constant weight 2^{m-1} (number of 1s), and the number of zeros is $2^{m-1} - 1$.

LFSR and Cyclic codes

The block of symbols forming a period of an m-sequence generated by a given LFSR can be considered (are) as codewords of a cyclic code which is the dual code

$$(2^m - 1, m, 2^{m-1})$$

of an Hamming code $(2^m - 1, 2^m - 1 - m, 3)$.
Every non-zero code word of a dual Hamming code has constant weight 2^{m-1} (number of 1s), and the number of zeros is $2^{m-1} - 1$.
This interpretation is useful for computing the run distribution within a codeword.

LFSR and Cyclic codes

LFSR and Cyclic codes

It also explains why their cyclic (or periodic) autocorrelation functions are ideal.

LFSR and Cyclic codes

It also explains why their cyclic (or periodic) autocorrelation functions are ideal.

Definition

The periodic autocorrelation function of a binary sequence $x(n)$ of length τ is defined as

$$c(\delta) = \frac{1}{\tau} \sum_{i=1}^{\tau} (-1)^{x(i+\delta)} (-1)^{x(i)}$$

LFSR and Cyclic codes

It also explains why their cyclic (or periodic) autocorrelation functions are ideal.

Definition

The periodic autocorrelation function of a binary sequence $x(n)$ of length τ is defined as

$$c(\delta) = \frac{1}{\tau}\sum_{i=1}^{\tau}(-1)^{x(i+\delta)}(-1)^{x(i)}$$

The aucocorrelation function of a binary m-sequence is

$$c(\delta) = \frac{1}{\tau}\sum_{i=1}^{\tau}(-1)^{x(i+\delta)+x(i)} = \begin{cases} 1 & \text{if } \delta = 0 \\ -\frac{1}{\tau} & \text{if } \delta \neq 0 \bmod \tau \end{cases}$$

Run distribution

A run of 1s of length k in a binary sequence consists of k consecutive 1s between two 0s

$$\ldots 01111110 \ldots \quad \ldots 0110 \quad \ldots 010 \quad \ldots 011111111110$$

and a run of 0s is similarly defined with the role of 0 and 1 exchanged.
Golomb derived the 0-1 run distributions, which are the same in any code word of a dual Hamming code:

- 1run of length m of '1s', and 0 runs of length m of '0s'
- 0 run of length $m-1$ of '1s', and 1 runs of length $m-1$ of '0s'
- 2^{m-k-2} runs of length k, of either '0s' or '1s',
  for $1 \leq k \leq m-2$.

(1)

LFSR: HW/SW complexity and classical structures

LFSR: HW/SW complexity and classical structures

The complexity of circuits or software programs realizing a LFSR can be defined as the number of additions in $\mathbb{F}_2$ required to produce an output bit.

LFSR: HW/SW complexity and classical structures

The complexity of circuits or software programs realizing a LFSR can be defined as the number of additions in $\mathbb{F}_2$ required to produce an output bit.
The complexity is essentially equal to the number of 1s in the matrix defining the LFSR.

LFSR: HW/SW complexity and classical structures

The complexity of circuits or software programs realizing a LFSR can be defined as the number of additions in $\mathbb{F}_2$ required to produce an output bit.
The complexity is essentially equal to the number of 1s in the matrix defining the LFSR.
Three structures are relevant

LFSR: HW/SW complexity and classical structures

The complexity of circuits or software programs realizing a LFSR can be defined as the number of additions in $\mathbb{F}_2$ required to produce an output bit.
The complexity is essentially equal to the number of 1s in the matrix defining the LFSR.
Three structures are relevant

- Fibonacci LFSR obtained from the companion matrix of $g(Z)$

LFSR: HW/SW complexity and classical structures

The complexity of circuits or software programs realizing a LFSR can be defined as the number of additions in $\mathbb{F}_2$ required to produce an output bit.
The complexity is essentially equal to the number of 1s in the matrix defining the LFSR.
Three structures are relevant

- Fibonacci LFSR obtained from the companion matrix of $g(Z)$
- Galois LFSR obtained from the transpose of the companion matrix of $g(Z)$

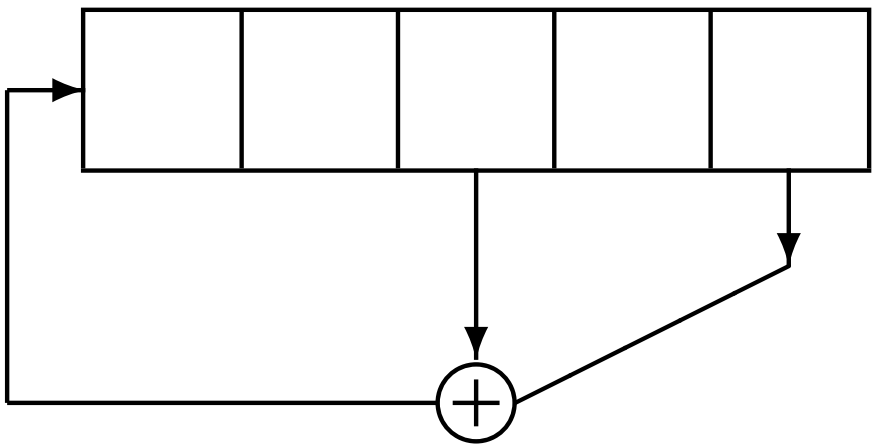
LFSR: HW/SW complexity and classical structures

The complexity of circuits or software programs realizing a LFSR can be defined as the number of additions in $\mathbb{F}_2$ required to produce an output bit.
The complexity is essentially equal to the number of 1s in the matrix defining the LFSR.
Three structures are relevant

- Fibonacci LFSR obtained from the companion matrix of $g(Z)$
- Galois LFSR obtained from the transpose of the companion matrix of $g(Z)$
- Tridiagonal LFSR obtained from a tridiagonal matrix with 1s in the upper and lower sub-diagonals.

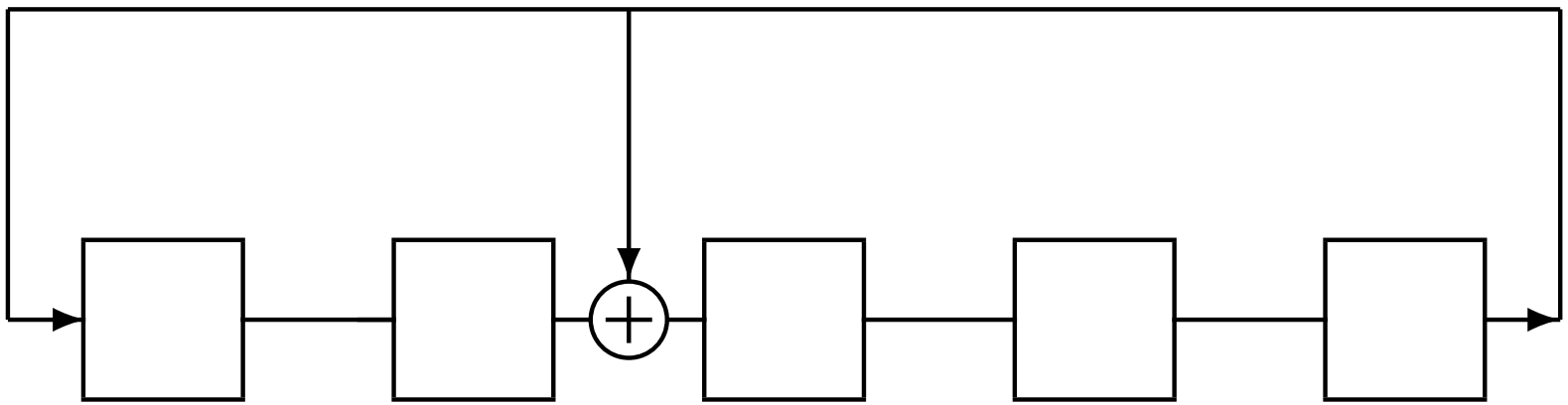
Fibonacci LFSR of order $m = 5$

$$\mathfrak{M}_F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$



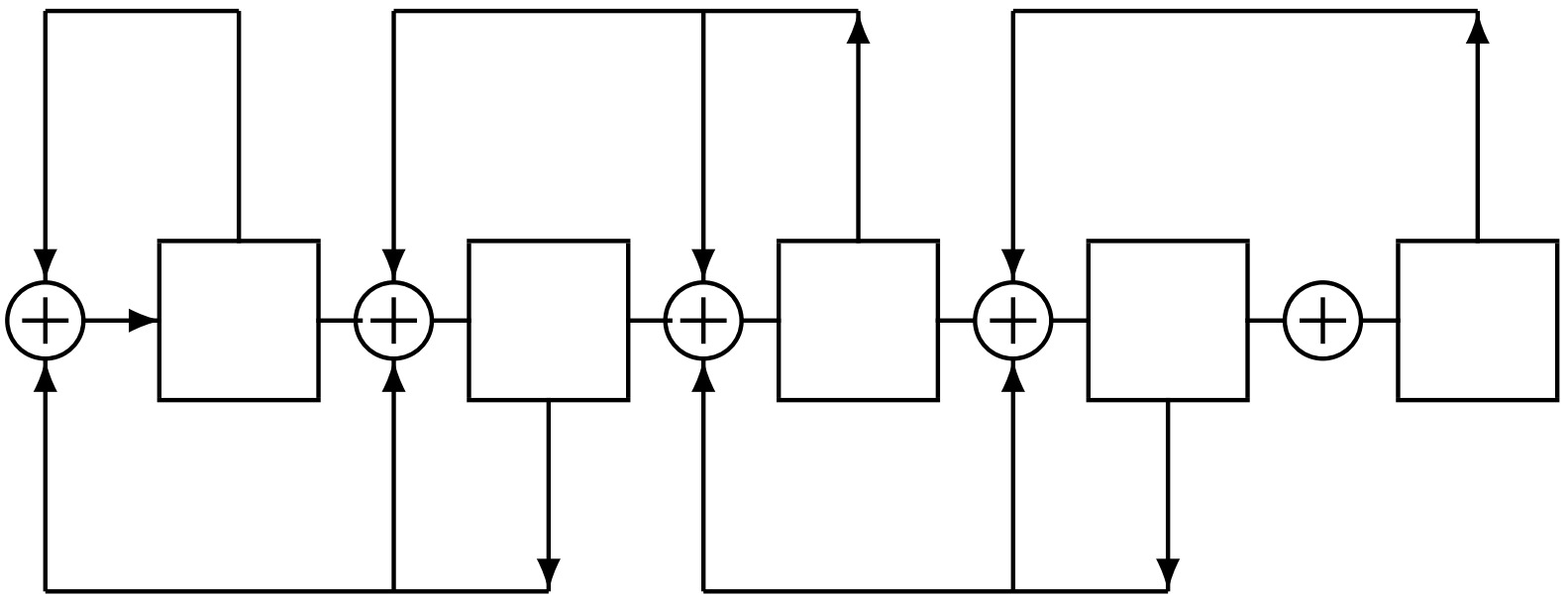
Galois LFSR of order $L = 5$

$$\mathfrak{M}_G = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Tridiagonal LFSR of order $L = 5$

$$\mathfrak{M}_T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



HW/SW complexity

- Fibonacci and Galois LFSRs have the same complexity, which is upper bounded by the length L.
  Note that, interesting generator polynomials have a small number of coefficients equal to 1s, possibly 3 coefficients .
  Unfortunately binary irreducible trinomials does not exist for every L.

HW/SW complexity

- Fibonacci and Galois LFSRs have the same complexity, which is upper bounded by the length L.
  Note that, interesting generator polynomials have a small number of coefficients equal to 1s, possibly 3 coefficients .
  Unfortunately binary irreducible trinomials does not exist for every L.
- Tridiagonal LFSR have a slightly larger complexity, which is upper bounded by $3L - 2$, nevertheless, in some circumstances may be preferred.

HW/SW complexity

- Fibonacci and Galois LFSRs have the same complexity, which is upper bounded by the length L.
  Note that, interesting generator polynomials have a small number of coefficients equal to 1s, possibly 3 coefficients .
  Unfortunately binary irreducible trinomials does not exist for every L.
- Tridiagonal LFSR have a slightly larger complexity, which is upper bounded by $3L - 2$, nevertheless, in some circumstances may be preferred.

HW/SW complexity

- Fibonacci and Galois LFSRs have the same complexity, which is upper bounded by the length L.
  Note that, interesting generator polynomials have a small number of coefficients equal to 1s, possibly 3 coefficients .
  Unfortunately binary irreducible trinomials does not exist for every L.
- Tridiagonal LFSR have a slightly larger complexity, which is upper bounded by $3L - 2$, nevertheless, in some circumstances may be preferred.

Note that not every binary polynomial of degree L is the characteristic polynomial of a tridiagonal matrix, however, it has been proved that every binary irreducible polynomial is the characteristic polynomial of a tridiagonal matrix.

Linear complexity profile

A linear sequence generated by a LFSR of length m has a period of length not greater than $2^m - 1$: that is "The linear complexity of the sequence is small with respect to its length".

Linear complexity profile

A linear sequence generated by a LFSR of length m has a period of length not greater than $2^m - 1$: that is "The linear complexity of the sequence is small with respect to its length".

Definition

The linear complexity $\ell(M)$ of a sequence $\mathcal{X}$ of length M is the minimum length of a LFSR that generates $\mathcal{X}$.

Linear complexity profile

A linear sequence generated by a LFSR of length m has a period of length not greater than $2^m - 1$: that is "The linear complexity of the sequence is small with respect to its length".

Definition

The linear complexity $\ell(M)$ of a sequence $\mathcal{X}$ of length M is the minimum length of a LFSR that generates $\mathcal{X}$.

If $\mathcal{X}$ is an m-sequence, then $\ell(M) = \lceil \log_2 M \rceil$

Linear complexity profile

A linear sequence generated by a LFSR of length m has a period of length not greater than $2^m - 1$: that is "The linear complexity of the sequence is small with respect to its length".

Definition

The linear complexity $\ell(M)$ of a sequence $\mathcal{X}$ of length M is the minimum length of a LFSR that generates $\mathcal{X}$.

If $\mathcal{X}$ is an m-sequence, then $\ell(M) = \lceil \log_2 M \rceil$

If $\mathcal{X}$ is a genuine random sequence, then $\ell(M) = \lceil \frac{M}{2} \rceil$

Berlekamp-Massey's algorithm

Given a sequence $\mathcal{X}$ of length N, the Berlekamp-Massey's algorithm yields the length of the shortest LFSR generating $\mathcal{X}$.

Berlekamp-Massey's algorithm

Given a sequence $\mathcal{X}$ of length N, the Berlekamp-Massey's algorithm yields the length of the shortest LFSR generating $\mathcal{X}$.

Question

Berlekamp-Massey's algorithm

Given a sequence $\mathcal{X}$ of length N, the Berlekamp-Massey's algorithm yields the length of the shortest LFSR generating $\mathcal{X}$.

Question

Which is the linear complexity of a genuine random sequence?

Berlekamp-Massey's algorithm

Given a sequence $\mathcal{X}$ of length N, the Berlekamp-Massey's algorithm yields the length of the shortest LFSR generating $\mathcal{X}$.

Question

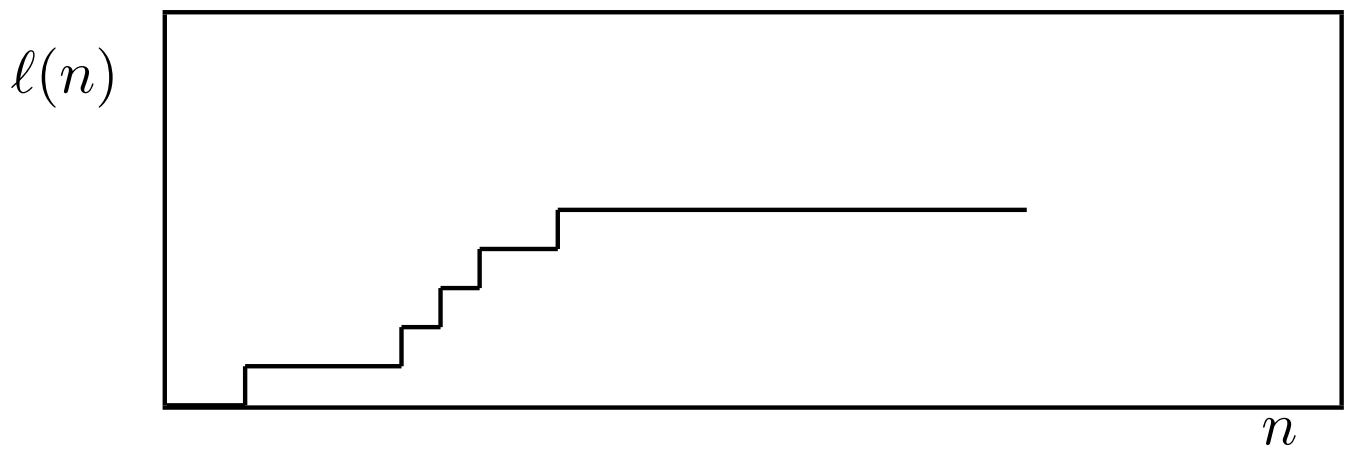
Which is the linear complexity of a genuine random sequence?
The approach is to compute for each subsequence of length n, for any n, its linear complexity, a task that yields the linear complexity profile.

Berlekamp-Massey's algorithm

Given a sequence $\mathcal{X}$ of length N, the Berlekamp-Massey's algorithm yields the length of the shortest LFSR generating $\mathcal{X}$.

Question

Which is the linear complexity of a genuine random sequence?
The approach is to compute for each subsequence of length n, for any n, its linear complexity, a task that yields the linear complexity profile.



Self-Clock Controlled LFSR

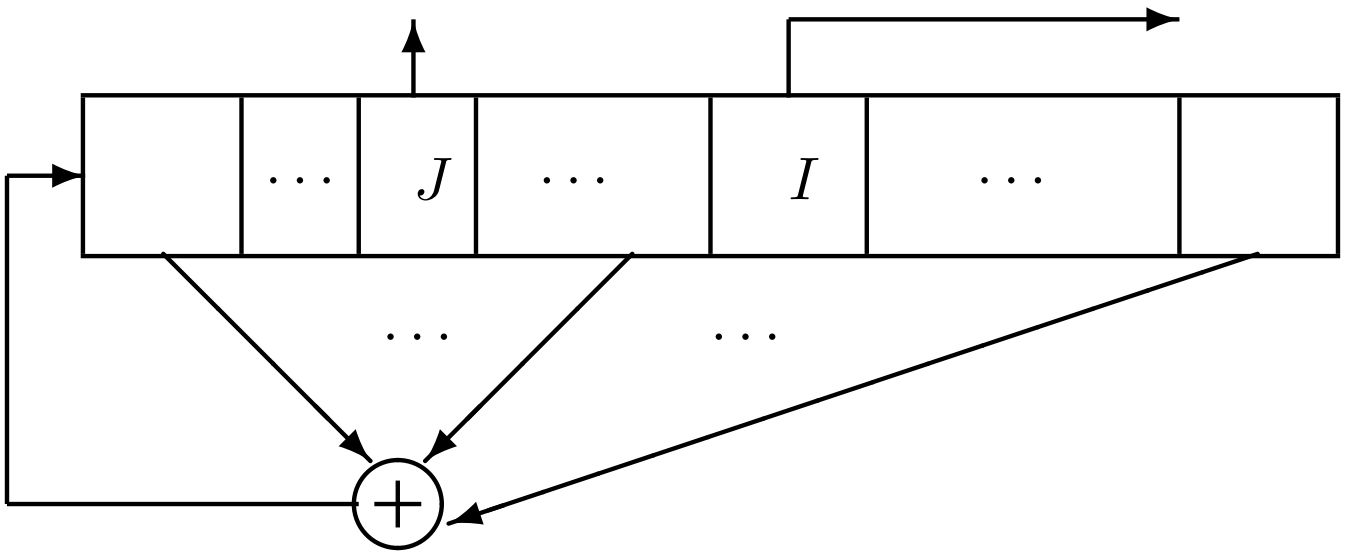
A self-clock controlled LFSR is a linear feedback shift register such that some states are skipped depending on the states themselves .

Practically some states are shadowed (hidden) for the external observer. The result is that it is difficult to predict from the generated sequence which are the skipped states

Self-Clock Controlled LFSR

A self-clock controlled LFSR is a linear feedback shift register such that some states are skipped depending on the states themselves .

Practically some states are shadowed (hidden) for the external observer. The result is that it is difficult to predict from the generated sequence which are the skipped states



Self-Clock Controlled Fibonacci LFSR

An example is a Fibonacci LFSR in which a cell J is marked: any time that, in the transition to a new state, in cell J occurs a 1, the new state is skipped and a second transition is operated (no further transition is done).

Example

00001	
10000	
01000	*skipped state*
00100	
10010	
01001	*skipped state*
10100	
01010	*skipped state*
00101	
00010	

Period of a Clock-controlled LFSR sequence

Period of a Clock-controlled LFSR sequence

- The period is approximately $2/3$ of τ:

$$\tau = \frac{2}{3}(2^m - 1) - \frac{2}{3}\delta = \frac{2}{3}\left(2^m - \frac{3 - (-1)^m}{2}\right) \quad .$$

Period of a Clock-controlled LFSR sequence

- The period is approximately 2/3 of τ:

$$\tau = \frac{2}{3}(2^m - 1) - \frac{2}{3}\delta = \frac{2}{3}\left(2^m - \frac{3 - (-1)^m}{2}\right) \quad .$$

- The generated sequences belong to a linear code: each sequence can be seen as a code word of a punctured code (the punctured symbols correspond to the skipped states).

Period of a Clock-controlled LFSR sequence

- The period is approximately 2/3 of τ:

$$\tau = \frac{2}{3}(2^m - 1) - \frac{2}{3}\delta = \frac{2}{3}\left(2^m - \frac{3 - (-1)^m}{2}\right) \quad .$$

- The generated sequences belong to a linear code: each sequence can be seen as a code word of a punctured code (the punctured symbols correspond to the skipped states).
- Different sequences produced by the same LFSR belong to different linear codes.

0-1 Distributions in clock-controlled LFSR sequences

The numbers N_{0I} and N_{1I} of '0s' and '1s', in a sequence generated by a self-clock controlled LFSR, depend on both the relative position of control and output cells, and the implementation LFSR-type, namely Fibonacci, Galois, or Tridiagonal. For the Fibonacci LFSR N_{0I} and N_{1I} can be computed in closed form.

Using the closed form of N_{0I} and N_{1I}, it is immediately seen that the clocked sequence is perfectly balanced, i.e. $N_{0I} = N_{1I}$, if and only if $I = 1$ if m is odd, and $I = 2$ if m is even.

Linear complexity profile

The Linear complexity profile of a clock controlled LFSR sequence is practically optimal as it can be theoretically shown

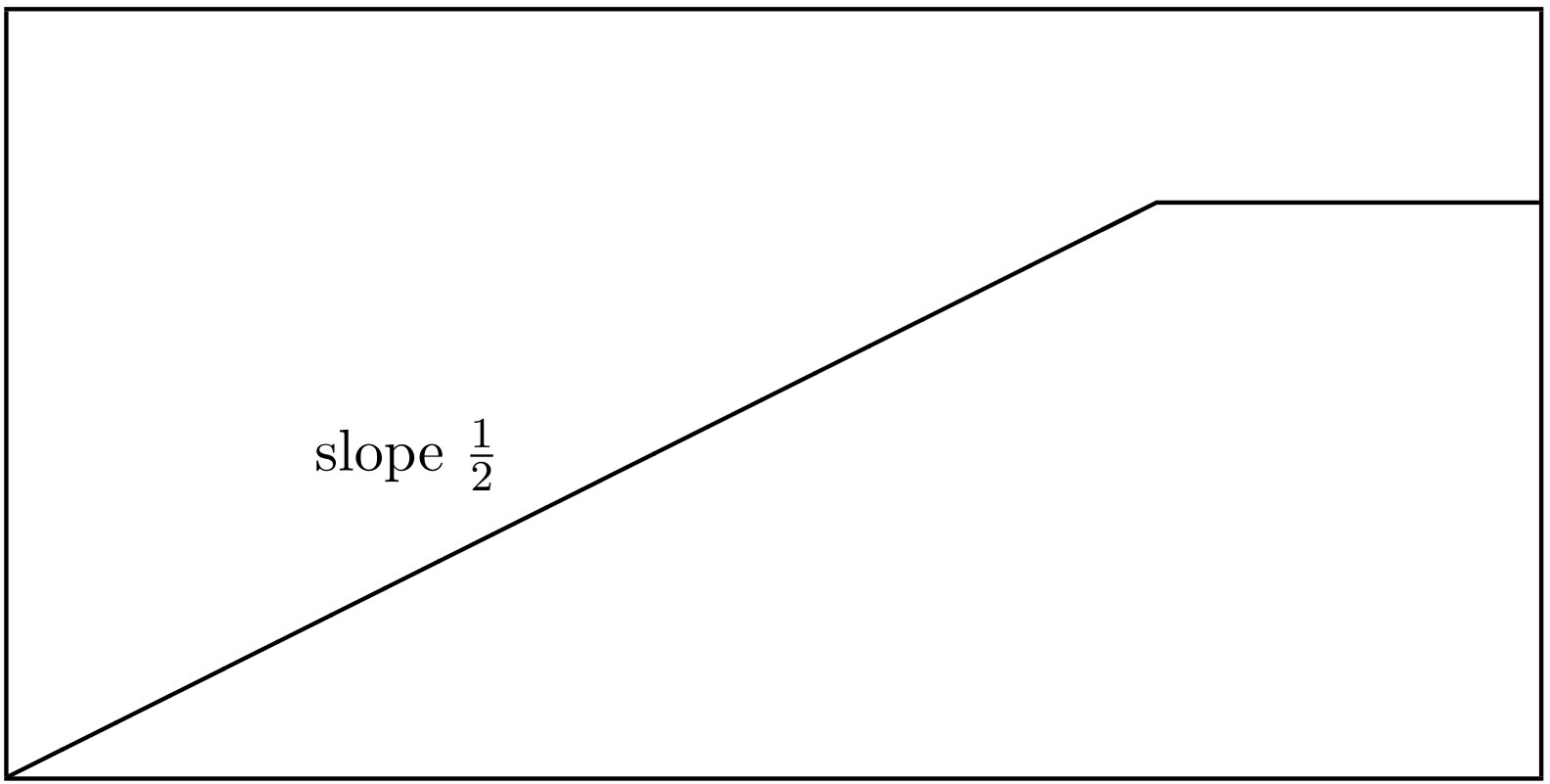


Figure: LCP for a clock controlled Fibonacci LFSR of length 22

Conclusions

Conclusions

1) LFSR are good generators of random sequences, which are easy to implement and may work fast.

Conclusions

1) LFSR are good generators of random sequences, which are easy to implement and may work fast.

Conclusions

1) LFSR are good generators of random sequences, which are easy to implement and may work fast.
   However, in cryptography, the use of these linear sequences needs further artifices to counteract the weaknesses implicit in the linearity.

Conclusions

1) LFSR are good generators of random sequences, which are easy to implement and may work fast.
   However, in cryptography, the use of these linear sequences needs further artifices to counteract the weaknesses implicit in the linearity.
2) Self-clock controlled LFSR have optimal linear complexity profile, indistinguishable from that of genuine random sequences, thus may be (more) directly used in cryptographic applications.

Conclusions

Conclusions

3) The observations collected in this talk have the modest aim of giving a quick view of the context in which is set the search for inexpensive mechanisms generating (binary) sequences that are good for cryptographic applications.

Conclusions

3) The observations collected in this talk have the modest aim of giving a quick view of the context in which is set the search for inexpensive mechanisms generating (binary) sequences that are good for cryptographic applications.

Conclusions

3) The observations collected in this talk have the modest aim of giving a quick view of the context in which is set the search for inexpensive mechanisms generating (binary) sequences that are good for cryptographic applications.

An endeavor that was and remains a source of challenging problems for engineers and mathematicians.

References

1. S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills, 1982.
2. D.E. Knuth, *The Art of Computer Programming,* Seminumerical algorithms, vol. II, Addison-Wesley, Reading Massachussetts, 1981.
3. R. Lidl, and H. Niederreiter, *Finite Fields*, Addison-Wesley, Reading, Mass., 1983.
4. J. Hoffstein, J. Pipher, J.H. Silverman, *An introduction to mathematical cryptography*, Springer, New York, 2008.

References

1. M. Elia, G. Morgari, M. Spicciola, On Binary Sequences Generated by Self-clock Controlled LFSR, MTNS 2010, Budapest, Hungary.
2. M. Elia, On Tridiagonal Binary Matrices and LFSRs, *Contemporary Eng. Sciences*, Vol. 3, no. 4, p167-182.
3. R.A. Rueppel, *Analysis and Design of Stream Cipher*, Springer, New York, 1986.
4. J.L. Massey, Shift-Register Synthesis and BCH decoding, *IEEE Trans. on Inform. Th.*, IT-15, 1969, pp.122-127.