Parallel Generation of ℓ-Sequences

Cédric Lauradoux[1] and Andrea Röck[2]

[1] Princeton University, Department of electrical engineering
Princeton, NJ 08544, USA
claurado@princeton.edu

[2] Team SECRET, INRIA Paris-Rocquencourt,
78153 Le Chesnay Cedex, France
andrea.roeck@inria.fr

Abstract. The generation of pseudo-random sequences at a high rate is an important issue in modern communication schemes. The representation of a sequence can be scaled by decimation to obtain parallelism and more precisely a sub-sequences generator. Sub-sequences generators and therefore decimation have been extensively used in the past for linear feedback shift registers (LFSRs). However, the case of automata with a non linear feedback is still in suspend. In this paper, we have studied how to transform of a feedback with carry shit register (FCSR) into a sub-sequences generator. We examine two solutions for this transformation, one based on the decimation properties of ℓ-sequences, *i.e.* FCSR sequences with maximal period, and the other one based on multiple steps implementation. We show that the solution based on the decimation properties leads to much more costly results than in the case of LFSRs. For the multiple steps implementation, we show how the propagation of carries affects the design.

Keywords: sequences, synthesis, decimation, parallelism, LFSRs, FCSRs.

1 Introduction

The synthesis of shift registers consists in finding the smallest automaton able to generate a given sequence. This problem has many applications in cryptography, sequences and electronics. The synthesis of a single sequence with the smallest linear feedback shift register is achieved by the Berlekamp-Massey [1] algorithm. There exists also an equivalent of Berlekamp-Massey in the case of multiple sequences [2,3]. In the case of FCSRs, we can use algorithms based on lattice approximation [4] or on Euclid's algorithm [5]. This paper addresses the following issue in the synthesis of shift registers: *given an automaton generating a sequence S, how to find an automaton which generates in parallel the sub-sequences associated to S*. Throughout this paper, we will refer to this problem as *the sub-sequences generator problem*. We aim to find the best solution to transform a 1-bit output pseudo-random generator into a multiple outputs generator. In particular, we investigate this problem when S is generated by a feedback with carry shit register (FCSR) with a maximal period, *i.e.* S is an ℓ-sequence. This class of pseudo-random generators was introduced by Klapper and Goresky in [6]. FCSRs and LFSRs are very similar in terms of properties [7,8]. However, FCSRs have a non-linear feedback which is a significant property to thwart algebraic attacks [9] in cryptographic applications [10].

The design of sub-sequences generators has been investigated in the case of LFSRs [11,12] and two solutions have been proposed. The first solution [13,14] is based on the classical synthesis of shift registers, *i.e.* the Berlekamp-Massey algorithm, to define each sub-sequence. The second solution [11] is based on a multiple steps design of the LFSR. We have applied those two solutions to FCSRs. The contributions of the paper are as follows:

- We explore the decimation properties of ℓ-sequences for the design of a sub-sequences generator by using an FCSR synthesis algorithm.
- We show how to implement a multiple steps FCSR in Fibonacci and Galois configuration.

The next section presents the motivation of this work and recalls the different representations of LFSRs and FCSRs. In Section 3, the existing results on LFSRs are described and we show multiple steps implementations of the Galois and the Fibonacci configuration. We describe in Section 4 our main results on the synthesis of sub-sequences generators in the case of ℓ-sequences. Then, we give some conclusions in Section 5.

2 Motivation and Preliminaries

The decimation is the main tool to transform a 1-bit output generator into a sub-sequences generator. This allows us to increase the throughput of a pseudo-random sequence generator (PRSG). Let $S = (s_0, s_1, s_2, \cdots)$ be an infinite binary sequence of period T, thus $s_j \in \{0, 1\}$ and $s_{j+T} = s_j$ for all $j \geq 0$. For a given integer d, a *d–decimation* of S is the set of sub-sequences defined by:

$$S_d^i = (s_i, s_{i+d}, s_{i+2d}, \cdots, s_{i+jd}, \cdots)$$

where $i \in [0, d-1]$ and $j = 0, 1, 2, \cdots$. Hence, a sequence S is completely described by the sub-sequences:

$$\begin{array}{ccc} S_d^0 & = & (s_0, s_d, \cdots) \\ S_d^1 & = & (s_1, s_{1+d}, \cdots) \\ \vdots & \vdots & \vdots \\ S_d^{d-2} & = & (s_{d-2}, s_{2d-2}, \cdots) \\ S_d^{d-1} & = & (s_{d-1}, s_{2d-1}, \cdots) \,. \end{array}$$

A single automaton is often used to generate the pseudo-random sequence S. In this case, it is difficult to achieve parallelism. The decomposition into sub-sequences overcomes this issue as shown by Lempel and Eastman in [11]. Each sub-sequence is associated to an automaton. Then, the generation of the d sub-sequences of S uses d automata which operate in parallel. Parallelism has two benefits, it can increase the throughput or reduce the power consumption of the automaton generating a sequence.

Throughput — The throughput $\mathcal{T}$ of a PRSG is defined by: $\mathcal{T} = n \times f$, with n is the number of bits produced every cycle and f is the clock frequency of the PRSG. Usually, we have $n = 1$, which is often the case with LFSRs. The decimation achieves a very interesting tradeoff for the throughput: $\mathcal{T}_d = d \times \gamma f$ with $0 < \gamma \leq 1$ the degradation factor of the original automaton frequency. The decimation provides an improvement of the throughput if and only if $\gamma d > 1$. It is then highly critical to find good automata for the generation of the sub-sequences. In an ideal case, we would have $\gamma = 1$ and then a d-decimation would imply a multiplication of the throughput by d.

Power consumption — The power consumption of a CMOS device can be estimated by the following equation: $P = C \times V_{dd}^2 \times f$, with C the capacity of the device and V_{dd} the supply voltage. The sequence decimation can be used to reduce the frequency of the device by interleaving the sub-sequences. The sub-sequences generator will be clocked at frequency $\frac{\gamma f}{d}$ and the outputs will be combined with a d-input multiplexer clocked at frequency γf. The original power consumption can then be reduced by the factor $\frac{\gamma}{d}$, where γ must be close to 1 to guarantee that the final representation of S is generated at frequency f.

The study of the γ parameter is out of the scope of this paper since it is highly related to the physical characteristics of the technology used for the implementation. In the following, we consider m-sequences and ℓ-sequences which are produced respectively by LFSRs and FCSRs.

Throughout the paper, we detail different representations of several automata. We denote by x_i a memory cell and by $(x_i)_t$ the content of the cell x_i at time t. The internal state of an automaton at time t is denoted by X_t.

2.1 LFSRs

An LFSR is an automaton which generates linear recursive sequences. A detailed description of this topic can be found in the monographs of Golomb and McEliece [15,16]. Let $s(x) = \sum_{i=0}^{\infty} s_i x^i$ define the power series of the sequence $S = (s_0, s_1, s_2, \ldots)$ produced by an LFSR. Then, there exists two polynomials, such that

$$s(x) = \frac{p(x)}{q(x)}$$

where $q(x)$ is the *connection polynomial* defined by the feedback positions of the automaton. Let m be the degree of $q(x)$, then the reciprocal polynomial $Q(x) = x^m q(1/x)$ is named the *characteristic polynomial.* An output sequence of an LFSR is called an *m–sequence* if it has the maximal period of $2^m - 1$. This is the case if and only if $q(x)$ is a primitive polynomial. There exists two different representations of an LFSR, the so-called Galois and Fibonacci setup,

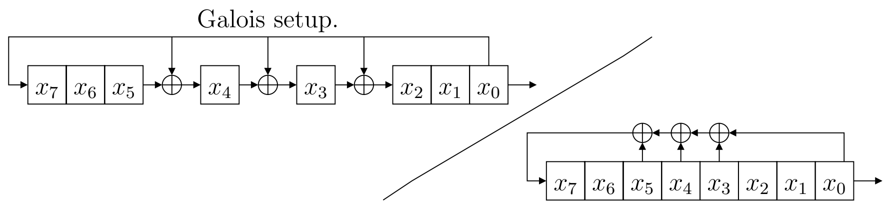


Fig. 1. Galois and Fibonacci LFSR

as we show in Figure 1. Both setups use the addition modulo 2. The Fibonacci setup is characterized by *a multiple inputs and single output feedback function*, while the Galois setup has *multiple feedback functions with a common input*. In both setups, we denote by x_0 the cell corresponding to the output of the LFSR. The same sequence can be produced by an LFSR in Fibonacci or Galois setup with the same characteristic polynomial but with different initializations. The *linear complexity* of a sequence S is defined as the size of the smallest LFSR which is able to produce this sequence [17].

2.2 FCSRs

FCSRs were introduced by Klapper and Goresky in [6]. Instead of addition modulo 2, FCSRs use additions with carry, which means that they need additional memory to store the carry. Their non–linear update function makes them particularly interesting for areas where linearity is an issue, like for instance stream ciphers. The output of a binary FCSR corresponds to the 2–adic expansion of the rational number:

$$\frac{h}{q} \leq 0 \; .$$

As for the LFSRs, there exists a Fibonacci and a Galois setup [7]. Their different structures can be seen in Figure 2, where $\sum$ represents the hamming weight of the incoming bits, $+$ an integer addition, c the additional memory for the carry, and $\boxplus$ an addition with carry of two bits where the carry is stored in $\square$. The

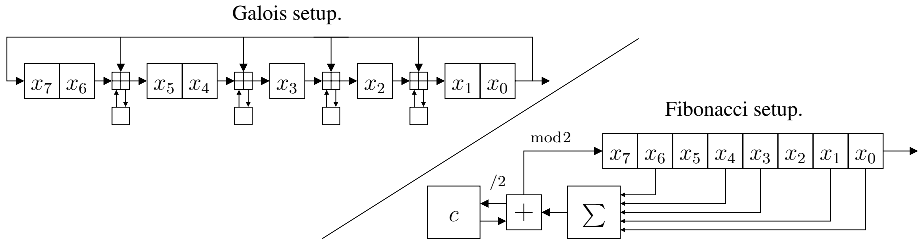

Galois setup.
x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
Fibonacci setup.
mod2
/2
c
$+$
$\sum$
x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0

Fig. 2. Galois and Fibonacci FCSR

value q determines the feedback positions of the automata, depending on the setup used. In this article, we consider *ℓ–sequences*, *i.e.* sequences with maximal period $\varphi(q)$, where φ denotes Euler's phi function. This is equivalent to the case that q is a prime power and 2 has multiplicative order $\varphi(q)$ modulo q. For simplicity reasons, we restrict ourselves also to the case where the generated sequence is strictly periodic. This property is equivalent to $-q \leq h \leq 0$, which is always the case for Galois FCSRs but not necessarily for Fibonacci FCSRs. The *2–adic complexity* [5] of a sequence is defined as the size of the smallest FCSR which produces this sequence. In the periodic case, this is equivalent to the bit length of q .

3 Sub-sequences Generators and m-Sequences

The decimation of LFSR sequences has been used in cryptography in the design of new stream ciphers [18]. There exists two approaches to use decimation theory to define the automata associated to the sub-sequences.

Construction using LFSR synthesis. This first solution associates an LFSR to each sub-sequence. It is based on well-known results on the decimation of LFSR sequences. It can be applied to both Fibonacci and Galois representation without any distinction.

Theorem 1 ([19,13]). *Let S be a sequence produced by an LFSR whose characteristic polynomial $Q(x)$ is irreducible in $\mathbf{F}_2$ of degree m. Let α be a root of $Q(x)$ and let T be the period of $Q(x)$. Let S_d^i be a sub-sequence resulting from the d-decimation of S. Then, S_d^i can be generated by an LFSR with the following properties:*

- *The minimum polynomial of α^d in $\mathbf{F}_{2^m}$ is the characteristic polynomial $Q^\star(x)$ of the resulting LFSR.*
- *The period $T^\star$ of $Q^\star(x)$ is equal to $\frac{T}{gcd(d,T)}$.*
- *The degree $m^\star$ of $Q^\star(x)$ is equal to the multiplicative order of $Q(x)$ in $\mathbf{Z}_{T^\star}$.*

In practice, the characteristic polynomial $Q^\star(x)$ can be determined using the Berlekamp-Massey algorithm [1]. The sub-sequences are generated using d LFSRs defined by the characteristic polynomial $Q^\star(x)$ but initialized with different values. In the case of LFSRs, the degree m^* must always be smaller or equal to m .

Construction using a multiple steps LFSR. This method was first proposed by Lempel and Eastman [11]. It consists in clocking the LFSR d times in one clock cycle by changing the connections between the memory cells and by some duplications of the feedback function. We obtain a network of linearly interconnected shift registers. This method differs for Galois and Fibonacci setup. The transformation of an m-bit Fibonacci LFSR into an automaton which generates d bits per cycle is achieved using the following equations:

$$next^d(x_i) = x_{i-d \bmod m} \tag{1}$$

$$(x_i)_{t+d} = \begin{cases} f(X_{t+i-m+d}) & \text{if } m-d \le i < m \\ (x_{i+d})_t & \text{if } i < m-d \end{cases} \tag{2}$$

where $next^d(x_i)$ is the cell connected to the output of x_i and f is the feedback function. The Equation 1 corresponds to the transformation of the connections between the memory cells. All the cells x_i of the original LFSR, such that i mod $d = k$, are gathered to form a sub-shift register, where $0 \le k \le d-1$. This is the basic operation to transform a LFSR into a sub-sequences generator with a multiple steps solution. The content of the last cell of the k-th sub-shift registers corresponds to the k-th sub-sequence S_d^k. The Equation 2 corresponds to the transformation of the feedback function. It must be noticed that the synthesis requires to have only relations between the state of the register at time $t+d$ and t. The Figure 3 shows an example of such a synthesis for a Fibonacci setup defined by the connection polynomial $q(x) = x^8 + x^5 + x^4 + x^3 + 1$ with the decimation factor $d = 3$. The transformation of a Galois setup is described by the Equations 1 and 3:

$$(x_i)_{t+d} = \begin{cases} (x_0)_{t+d-m+i} \oplus \bigoplus_{k=0}^{m-2-i} a_{i+k}\,(x_0)_{t+d-k-1} & \text{if } m-d \le i < m \\ (x_{i+d})_t \oplus \bigoplus_{k=0}^{d-1} a_{i+d-1-k}\,(x_0)_{t+k} & \text{if } i < m-d \end{cases} \tag{3}$$

with $q(x) = 1 + a_0x + a_1x^2 + \cdots + a_{m-2}x^{m-1} + x^m$. The Equation 3 does not provide a direct relation between the state of the register at time $t+d$ and t. However, this equation can be easily derived to obtain more practical formulas as shown in Figure 4.

$$\begin{aligned} &next^1(x_0) = x_7 \\ &next^1(x_i) = x_{i-1} \text{ if } i \neq 0 \\ &(x_7)_{t+1} = (x_3)_t \oplus (x_4)_t \oplus (x_5)_t \oplus (x_0)_t \\ &(x_i)_{t+1} = (x_{i+1})_t \text{ if } i \neq 7 \end{aligned}$$

$$\begin{aligned} &next^3(x_0) = x_5 \\ &next^3(x_1) = x_6 \\ &next^3(x_2) = x_7 \\ &next^3(x_i) = x_{i-3} \text{ if } i > 2 \\ &(x_5)_{t+3} = (x_3)_t \oplus (x_4)_t \oplus (x_5)_t \oplus (x_0)_t \\ &(x_6)_{t+3} = (x_4)_t \oplus (x_5)_t \oplus (x_6)_t \oplus (x_1)_t \\ &(x_7)_{t+3} = (x_5)_t \oplus (x_6)_t \oplus (x_7)_t \oplus (x_2)_t \\ &(x_i)_{t+3} = (x_{i+3})_t \text{ if } i < 5 \end{aligned}$$

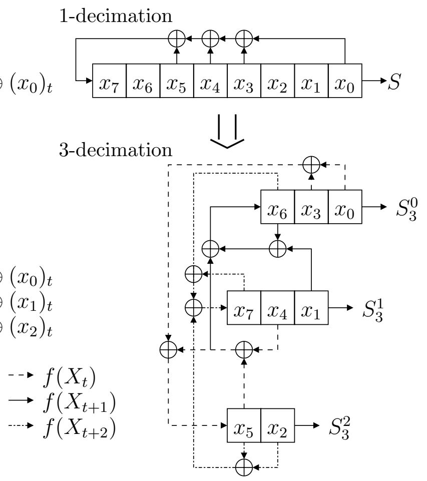


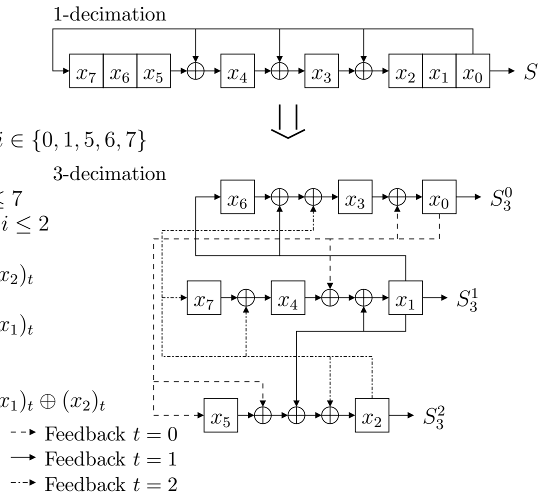
Fig. 3. Multiple steps generator for a Fibonacci LFSR

Table 1. Comparison of the two methods for the synthesis of a sub-sequences generator

Method	Memory cells	Logic Gates
LFSR synthesis	$d \times m^\star$	$d \times wt(Q^\star)$
Multiple steps LFSRs [11]	m	$d \times wt(Q)$

$$
\begin{aligned}
&next^1(x_i) = x_{i-1} \text{ if } i \neq 0\\
&next^1(x_0) = x_7\\
&(x_2)_{t+1} = (x_3)_t \oplus (x_0)_t\\
&(x_3)_{t+1} = (x_4)_t \oplus (x_0)_t\\
&(x_4)_{t+1} = (x_5)_t \oplus (x_0)_t\\
&(x_i)_{t+1} = (x_{i+1 \pmod m})_t, \forall i \in \{0,1,5,6,7\}
\end{aligned}
$$

$$
\begin{aligned}
&next^3(x_i) = x_{i-3} \text{ if } 2 < i \leq 7\\
&next^3(x_i) = x_{m+i-d} \text{ if } 0 \leq i \leq 2\\
&(x_0)_{t+3} = (x_3)_t \oplus (x_0)_t\\
&(x_3)_{t+3} = (x_6)_t \oplus (x_1)_t \oplus (x_2)_t\\
&(x_6)_{t+3} = (x_1)_t\\
&(x_1)_{t+3} = (x_4)_t \oplus (x_0)_t \oplus (x_1)_t\\
&(x_4)_{t+3} = (x_2)_t \oplus (x_7)_t\\
&(x_7)_{t+3} = (x_2)_t\\
&(x_2)_{t+3} = (x_5)_t \oplus (x_0)_t \oplus (x_1)_t \oplus (x_2)_t\\
&(x_5)_{t+3} = (x_0)_t
\end{aligned}
$$



1-decimation
x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
S
3-decimation
x_6 x_3 x_0 S_3^0
x_7 x_4 x_1 S_3^1
x_5 x_2 S_3^2
Feedback $t = 0$
Feedback $t = 1$
Feedback $t = 2$

Fig. 4. Multiple steps generator for a Galois LFSR

Comparison. We have summarized in the Table 1 the two methods used to synthesize the sub-sequences generator. By $wt(Q(x))$, we mean the Hamming weight of Q, *i.e.* the number of non-zero monomials. The method based on LFSR synthesis proves that there exists a solution for the synthesis of the sub-sequences generator. With this solution, both memory cost and gate number depends on the decimation factor d. The method proposed by Lempel and Eastman [11] uses a constant number of memory cells for the synthesis of the sub-sequences generator.

The sub-sequences generators defined with the Berlekamp-Massey algorithm are not suitable to reduce the power consumption of an LFSR. Indeed, d LFSRs will be clocked to produce the sub-sequences, however the power consumption of such a sub-sequence generator is given by:

$$
\begin{aligned}
P &= d \times \left(C_d \times V_{dd}^2 \times \frac{\gamma f}{d} \right)\\
&= \lambda C \times V_{dd}^2 \times \gamma f
\end{aligned}
$$

with $C_d = \lambda C$ and C the capacity of the original LFSR. We can achieve a better result with a multiple steps LFSR:

$$
P = \lambda' C \times V_{dd}^2 \times \frac{\gamma f}{d}
$$

with $C_d = \lambda' C$.

4 Sub-sequences Generators and ℓ-Sequences

This section presents the main contribution of the paper. We apply the two methods used in the previous section on the case of ℓ-sequences.

Construction using FCSR synthesis. There exists algorithms based on Euclid's algorithm [5] or on lattice approximation [4], which can determine the smallest FCSR to produce S_d^i. These algorithms use the first k bits of S_d^i to find $h^\star$ and $q^\star$ such that $h^\star/q^\star$ is the 2-adic representation of the sub-sequence, $-q^\star < h^\star \leq 0$ and $\gcd(q^\star, h^\star) = 1$. Subsequently, we can find the feedback positions and the initial state of the FCSR in Galois or Fibonacci architecture. The value k is in the range of twice the linear 2-adic complexity of the sequence. For our new sequence S_d^i, let $h^\star$ and $q^\star$ define the values found by one of the algorithms mentioned above. By $T^\star$ and T, we mean the periods of respectively S_d^i and S.

For the period of the decimated sequences, we can make the following statement, which is true for all periodic sequences.

Lemma 1. *Let $S = (s_0, s_1, s_2, \ldots)$ be a periodic sequence with period T. For a given $d > 1$ and $0 \leq i \leq d-1$, let S_d^i be the decimated sequence with period $T^\star$. Then, it must hold:*

$$T^\star \left| \frac{T}{\gcd(T,d)} \right. \quad . \tag{4}$$

If $\gcd(T,d) = 1$ *then* $T^\star = T$.

Proof. The first property is given by:

$$s_{d[j+T/gcd(T,d)]+i} = s_{dj+i+T[d/\gcd(T,d)]} = s_{dj+i} \ .$$

For the case $\gcd(T,d) = 1$, there exits $x, y \in \mathbb{Z}$ such that $xT + yd = 1$ due to Bezout's lemma. Since S is periodic, we define for any $j < 0$ and $k \geq 0$, $s_j = s_k$ such that $j \pmod T = k$. Thus, we can write for any j:

$$\begin{aligned} s_j &= s_{i+(j-i)xT+(j-i)yd} &&= s_{i+(j-i)yd} \ , \\ s_{j+T^\star} &= s_{i+(j-i)xT+T^\star xT+(j-i)yd+T^\star yd} &&= s_{i+(j-i)yd+T^\star yd} \ . \end{aligned}$$

However, since $T^\star$ is the period of S_d^i we get:

$$s_{j+T^\star} = s_{j+(j-i)yd} = s_j \ .$$

Therefore, it must hold that $T | T^\star$ which together with (4) proves that $T^\star = T$ if $\gcd(T,d) = 1$. □

In the case of $\gcd(T,d) > 1$, *the real value of $T^\star$ might depend on i, e.g.* for S being the 2-adic representation of $-1/19$ and $d = 3$ we have $T/gcd(T,d) = 6$, however, for S_3^0 the period $T^\star = 2$ and for S_3^1 the period $T^\star = 6$.

A critical point in this approach is that the size of the new FCSR can be exponentially bigger than the original one. In general, we only know that for the new $q^\star$ it must hold that $q^\star | 2^{T^\star} - 1$. From the previous paragraph we know that $T^\star$ can be as big as $T/gcd(T,d)$. In the case of an *allowable decimation* [20], *i.e.* a decimation where d and T are coprime, we have more informations:

Corollary 1 ([21]). *Let S be the 2-adic representation of h/q, where $q = p^e$ is a prime power with p prime, $e \geq 1$ and $-q \leq h \leq 0$. Let $d > 0$ be relatively prime to $T = p^e - p^{e-1}$, the period of S. Let S_d^i be a d-decimation of S with $0 \leq i < d$ and let $h^\star/q^\star$ be its 2-adic representation such that $-q^\star \leq h^\star \leq 0$ and* $\gcd(q^\star, h^\star) = 1$*. Then $q^\star$ divides*

$$2^{T/2} + 1 \, .$$

If the following conjecture is true, we have more information on the new value $q^\star$.

Conjecture 1 ([21]). Let S be an ℓ-sequence with connection number $q = p^e$ and period T. Suppose p is prime and $q \notin \{5, 9, 11, 13\}$. Let d_1 and d_2 be relatively prime to T and incongruent modulo T. Then for any i and j, $S_{d_1}^i$ and $S_{d_2}^j$ are cyclically distinct, *i.e.* there exists no $\tau \geq 0$ such that $S_{d_1}^i$ is equivalent to $S_{d_2}^j$ shifted by τ positions to the left.

This conjecture has already been proved for many cases [20,22] but not yet for all. If it holds, this implies that for any $d > 1$, S_d^i is cyclically distinct from our original ℓ-sequence. We chose q such that the period was maximal, thus, any other sequence with the same period which is cyclically distinct must have a value $q^\star > q$. This means that the complexity of the FCSR producing the subsequence S_d^i will be larger than the original FCSR, *if d and T are relative prime.*

Remark 1. Let us consider the special case where q is prime and the period $T = q - 1 = 2p$ is twice a prime number $p > 2$, as recommended for the stream cipher proposed in [23]. The only possibilities in this case for $gcd(d, T) > 1$ is $d = 2$ or $d = T/2$.

For $d = T/2$, we will have $T/2$ FCSRs where each of them outputs either $0101\ldots$ or $1010\ldots$, since for an ℓ-sequence the the second half of the period is the inverse of the first half [21]. Thus, the size of the sub-sequences generator will be in the magnitude of T which is exponentially bigger than the 2-adic complexity of S which is $\log_2(q)$.

In the case of $d = 2$, we get two new sequences with period $T^\star = p$. As for any FCSR, it must hold that $T^\star | ord_{q^\star}(2)$, where $ord_{q^\star}(2)$ is the multiplicative order of 2 modulo $q^\star$. Let $\varphi(n)$ denote Euler's function, *i.e.* the number of integers smaller than n which are relative prime to n. It is well known, *e.g.* [16], that $ord_{q^\star}(2) | \varphi(q^\star)$ and if $q^\star$ has the prime factorization $p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ then $\varphi(q^\star) = \prod_{i=1}^{r} p_i^{e_i - 1}(p_i - 1)$. From this follows that $p \neq \varphi(q^\star)$, because otherwise $(p+1)$ must be a prime number, which is not possible since $p > 2$ is a prime. We also know that $T^* = p | \varphi(q^*)$, thus $2 \times p = q - 1 \leq \varphi(q^*)$. This implies that $q^\star > q$, since from $T^\star = T/2$ follows that $q \neq q^\star$.

Together with Conjecture 1, we obtain that for such an FCSR any decimation would have a larger complexity than the original one. This is also interesting from the perspective of stream ciphers, since any decimated subsequence of such an FCSR has larger 2-adic complexity than the original one, except for the trivial case with $d = T/2$.

Construction using a multiple steps FCSR. A multiple steps FCSR is a network of interconnected shift registers with a carry path: the computation of the feedback at time t depends directly on the carry generated at time $t-1$. The transformation of an m-bit FCSR into a d sub-sequences generator uses first Equation 1 to modify the mapping of the shift register. For the Fibonacci setup, the transformation uses the following equations:

$$(x_i)_{t+d} = \begin{cases} g\left(X_{t+d-m+i}, c_{t+d-m+i}\right) \bmod 2 & \text{if } m-d \le i < m \\ (x_{i+d})_t & \text{if } i < m-d \end{cases} \tag{5}$$

$$c_{t+d} = g(X_{t+d-1}, c_{t+d-1})/2 \tag{6}$$

with $g(X_t, c_t) = h(X_t) + c_t$ the feedback function of an FCSR in Fibonacci setup and

$$h(X_t) = \sum_{i=0}^{m-1} a_i (x_i)_t \ . \tag{7}$$

Due to the nature of the function g, we can split the automaton into two parts. The first part handles the computation related to the shift register X_t and the other part is the carry path as shown in Figure 5 for $q = 347$.

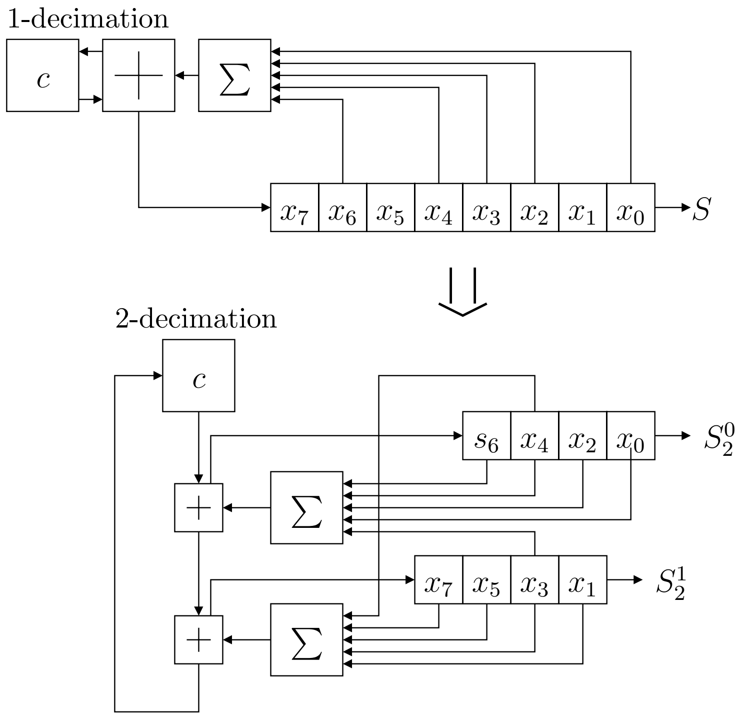
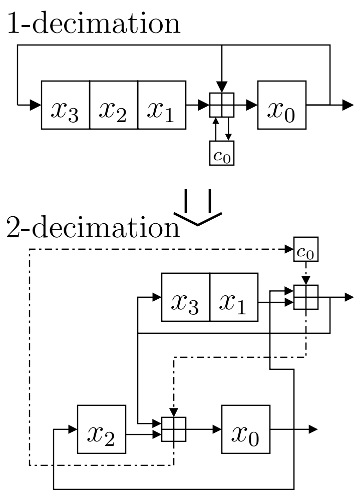
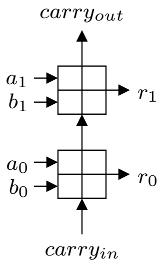


Fig. 5. Multiple steps generator for a Fibonacci FCSR



(a) Galois FCSR for $q = 19$.



(b) 2-bit ripple carry adder.

Fig. 6. Example for a Galois FCSR with $q = 19$

The case of Galois FCSRs is more difficult because the circuit can not be split into two parts: each bit of the carry register must be handle separately. The modification of the basic operator of a Galois FCSR, *i.e.* addition with carry, is the key transformation to obtain a sub-sequences generator. Let us consider a Galois FCSR with $q = 19$. This automaton has a single addition with carry as shown in Figure 6(a). The sub-sequences generator for $d = 2$ associated to this FCSR is defined by:

$$t+1 \begin{cases} (x_0)_{t+1} = (x_0)_t \oplus (x_1)_t \oplus (c_0)_t \\ (c_0)_{t+1} = [(x_0)_t \oplus (x_1)_t]\,[(x_0)_t \oplus (c_0)_t] \oplus (x_0)_t \end{cases} \tag{8}$$

$$t+2 \begin{cases} (x_0)_{t+2} = (x_0)_{t+1} \oplus (x_2)_t \oplus (c_0)_{t+1} \\ (c_0)_{t+2} = [(x_0)_{t+1} \oplus (x_2)_t]\,[(x_0)_{t+1} \oplus (c_0)_{t+1}] \oplus (x_0)_{t+1} \end{cases} \tag{9}$$

with c_0 the carry bit of the FCSR. The previous equations correspond to the description of the addition with carry at the bit-level (and represented by $\boxplus$ in the figures). This operator is also known as a full adder. The Equations corresponding to time $t+2$ depend on the carry, $(c_0)_{t+1}$, generated at time $t+1$. This dependency between full adders is a characteristic of a well-known arithmetic circuit: the n-bit ripple carry adder (Figure 6(b)).

Thus, all the full adders in a d sub-sequences generator are replaced by d-bit ripple carry adders as shown in Figure 7.

We can derive a more general representation of a multiple steps Galois FCSR from the previous formula:

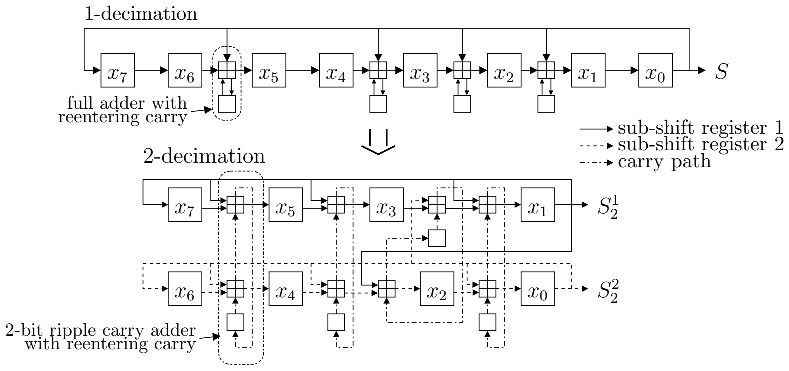


Fig. 7. Multiple steps generator for a Galois FCSR

$$(x_i)_{t+d} = \begin{cases} (x_0)_{t+d-m+i} \oplus \bigoplus_{k=0}^{m-2-i} a_{i+k}\left[(x_0)_{t+d-k-1} \oplus (c_{i+k})_{t+d-k-1}\right] \\ \hfill \text{if } m-d \le i < m \\ (x_{i+d})_t \oplus \bigoplus_{k=0}^{d-1} a_{i+k}\left[(x_0)_{t+d-k-1} \oplus (c_{i+k})_{t+d-k-1}\right] \\ \hfill \text{if } i < m-d \end{cases} \tag{10}$$

$$(c_i)_{t+d} = \left[(x_0)_{t+d-1} \oplus (x_{i+1})_{t+d-1}\right]\left[(x_0)_{t+d-1} \oplus (c_i)_{t+d-1}\right] \oplus (x_0)_{t+d-1} \tag{11}$$

The Equation 11 shows the dependencies between the carries which corresponds to the propagation of the carry in a ripple carry adder. The Equation 10 corresponds to the path of the content of a memory cell $(x_i)_t$ through the different levels of the ripple carry adders.

Comparison The construction using FCSR synthesis is more complicated than in the case of LFSRs. The size of the minimal generator producing S_d^i can depend on i, and we can only give upper bounds for $q^\star$ namely that $q^\star | 2^{T^\star} - 1$ in the general case and that $q^\star | 2^{T/2} + 1$ if $\gcd(d, T) = 1$. Based on Conjecture 1, we saw that $q^\star > q$ if $\gcd(T, d) = 1$. Moreover, in the case $p = 2p + 1$ with p and q prime, the resulting sub-sequences generator is always larger than the original one.

Apart from the carry path and the cost of the addition with carry, the complexity of a multiple steps implementation of a FCSR is very similar to the multiple steps implementation of an LFSR. There is no overhead in memory and the number of logic gates for a Galois FCSR is $5d \times wt(q)$ where $wt(q)$ is the number of ones in the binary representation of q and the number 5 corresponds to the five gates required for a full-adder (four Xors and one And *cf.* Equation 8). In the case of Fibonacci setup, the number of logic gates is given by:

$$\begin{aligned} &d \times (5 \times (wt(q) - \lceil \log_2(wt(q)+1) \rceil) + \\ &\qquad +2 \times (\lceil \log_2(wt(q)+1) \rceil - wt(wt(q)) + 5 \times size(c)) \end{aligned}$$

with $(5 \times (wt(q) - \lceil \log_2(wt(q)+1) \rceil) + 2 \times (\lceil \log_2(wt(q)+1) \rceil - wt(wt(q)))$ the cost of the implementation of a parallel bit counter [24], *i.e.* the h function (*cf.*

Equation 7) and $5 \times size(c)$ is the cost of a ripple carry adder which adds the content of $size(c)$ bits of the carry with the output of h.

5 Conclusion

We have presented in this paper how to transform an FCSR into a sub-sequences generator to increase its throughput or to reduce its power consumption. Our results emphasize the similarity between LFSRs and FCSRs in terms of implementation properties. In both cases, the solution based on the classical synthesis algorithm fails to provide a minimal solution. Even worse, in the case of FCSRs the memory needed is in practice exponentially bigger than for the original FCSR. Thus, we need to use multiple steps implementations as for LFSRs. The propagation of carries is the main problem in multiple steps implementations of FCSRs: if we consider d sub-sequences, we obtain d-bit ripple carry adders with carry instead of additions with carry in a Galois setup. In the case of a Fibonacci setup, the situation is different since we can split the feedback function into two parts. This new representation can significantly improve the hardware implementation of FCSRs but it may also be possible to improve software implementations.

References

1. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Transactions on Information Theory 15, 122–127 (1969)
2. Feng, G.L., Tzeng, K.: A Generalization of the Berlekamp-Massey Algorithm for Multisequence Shift-Register Synthesis with Applications to Decoding Cyclic Codes. IEEE Transactions on Information Theory 37(5), 1274–1287 (1991)
3. Schmidt, G., Sidorenko, V.: Linear Shift-Register Synthesis for Multiple Sequences of Varying Length. In: IEEE International Symposium on Information Theory - ISIT 2006, pp. 1738–1742. IEEE, Los Alamitos (2006)
4. Klapper, A., Goresky, M.: Feedback shift registers, 2-adic span, and combiners with memory. Journal of Cryptology 10, 111–147 (1997)
5. Arnault, F., Berger, T.P., Necer, A.: Feedback with Carry Shift Registers synthesis with the Euclidean Algorithm. IEEE Transactions on Information Theory 50(5) (2004)
6. Klapper, A., Goresky, M.: 2-adic shift registers. In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 174–178. Springer, Heidelberg (1994)
7. Goresky, M., Klapper, A.: Fibonacci and Galois representations of feedback-with-carry shift registers. IEEE Transactions on Information Theory 48(11) (2002)
8. Goresky, M., Klapper, A.: Algebraic Shift Register Sequences (preprint)
9. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
10. Arnault, F., Berger, T.P., Minier, M.: On the security of FCSR-based pseudorandom generators. In: State of the Art of Stream Ciphers - SASC (2007), http://sasc.crypto.rub.de/program.html

11. Lempel, A., Eastman, W.L.: High Speed Generation of Maximal Length Sequences. IEEE Transactions on Computer 2, 227–229 (1971)
12. Smeets, B.J.M., Chambers, W.G.: Windmill Generators: A Generalization and an Observation of How Many There Are. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 325–330. Springer, Heidelberg (1988)
13. Rueppel, R.A.: Analysis and Design of Stream Ciphers. Springer, Heidelberg (1986)
14. Filiol, E.: Decimation Attack of Stream Ciphers. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 31–42. Springer, Heidelberg (2000)
15. Golomb, S.W.: Shift Register Sequences. Aegean Park Press (1981)
16. McEliece, R.J.: Finite field for scientists and engineers. Kluwer Academic Publishers, Dordrecht (1987)
17. Cusick, T.W., Ding, C., Renvall, A.: Stream Ciphers and Number Theory. North-Holland, Amsterdam (1998)
18. Massey, J.L., Rueppel, R.A.: Linear ciphers and random sequence generators with multiple clocks. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 74–87. Springer, Heidelberg (1985)
19. Zierler, N.: Linear recurring Sequences. Journal of the Society for Industrial and Applied Mathematics 2, 31–48 (1959)
20. Goresky, M., Klapper, A., Murty, R., Shparlinski, I.: On Decimations of ℓ-Sequences. SIAM Journal of Discrete Mathematics 18(1), 130–140 (2004)
21. Goresky, M., Klapper, A.: Arithmetic crosscorrelations of feedback with carry shift register sequences. IEEE Transactions on Information Theory 43(4), 1342–1345 (1997)
22. Xu, H., Qi, W.: Further Results on the Distinctness of Decimations of ℓ-Sequences. IEEE Transactions on Information Theory 52(8), 3831–3836 (2006)
23. Arnault, F., Berger, T.P.: F-fcsr: Design of a new class of stream ciphers. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 83–97. Springer, Heidelberg (2005)
24. Muller, D.E., Preparata, F.P.: Bounds to complexities of networks for sorting and switching. J. ACM 22, 1531–1540 (1975)
25. Hurd, W.: Efficient Generation of Statistically Good Pseudonoise by Linearly Interconnected Shift Registers. IEEE Transactions on Computer 2, 146–152 (1974)
26. Rueppel, R.A.: When Shift Registers Clock Themselves. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 53–64. Springer, Heidelberg (1988)
27. Key, E.L.: An Analysis of the Structure and Complexity of Nonlinear Binary Sequence Generators. IEEE Transactions Information Theory 22(4), 732–736 (1976)
28. Berger, T.P., Minier, M.: Two Algebraic Attacks Against the F-FCSRs Using the IV Mode. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 143–154. Springer, Heidelberg (2005)
29. Arnault, F., Berger, T.P.: Design and Properties of a New Pseudorandom Generator Based on a Filtered FCSR Automaton. IEEE Transaction on Computers. 54(11), 1374–1383 (2005)