Parallel Generation of ℓ Sequences (extended abstract)

Cédric Lauradoux*and Andrea Röck†

Dagstuhl Seminar 09031, January 2009

1 Introduction

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 work, we study how to transform a feedback with carry shift 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.

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 [Mas69] algorithm. There exists also an equivalent of Berlekamp-Massey in the case of multiple sequences [FT91, SS06]. In the case of FCSRs, we can use algorithms based on lattice approximation [KG97] or on Euclid's algorithm [ABN04]. This work addresses the following issue in the synthesis of shift registers: *given an automaton generating a sequence $\mathcal{S}$, how to find an automaton which generates in parallel the sub-sequences associated to $\mathcal{S}$.* 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 $\mathcal{S}$ is generated by a feedback with carry shift register (FCSR) with a maximal period, *i.e.* $\mathcal{S}$ is an ℓ-sequence.

The design of sub-sequences generators has been investigated in the case of LFSRs [LE71, SC88] and two solutions have been proposed. The first solution [Rue86, Fil00] is based on the classical synthesis of shift registers, *i.e.* the Berlekamp-Massey algorithm, to define each sub-sequence. The second solution [LE71] is based on a multiple steps design of the LFSR. We have applied those two solutions to FCSRs. Our contributions 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.

*Princeton University, Department of electrical engineering, Princeton, NJ 08544, USA
†INRIA Paris-Rocquencourt, Project-Team SECRET, 78153 Le Chesnay Cedex, France

Dagstuhl Seminar Proceedings 09031
Symmetric Cryptography
http://drops.dagstuhl.de/opus/volltexte/2009/1956

- We show how to implement a multiple steps FCSR in Fibonacci and Galois configuration.

This work was presented at the international conference on SEquences and Their Applications (SETA) 2008 [LR08].

2 Motivation

The decimation is the main tool to transform a 1-bit output generator into a sub-sequences generator. An example for a 2–decimation can be found in Figure 1. The decimation allows us to

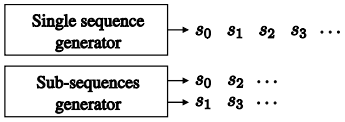


Figure 1: Model of a sub-sequences generator.

increase the throughput of a pseudo-random sequence generator (PRSG). Let $\mathcal{S} = (s_0, s_1, s_2, \cdots)$ 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 $\mathcal{S}$ is the set of sub-sequences defined by:

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

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

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

A single automaton is often used to generate the pseudo-random sequence $\mathcal{S}$. In this case, it is difficult to achieve parallelism. The decomposition into sub-sequences overcomes this issue as shown by Lempel and Eastman in [LE71]. Each sub-sequence is associated to an automaton. Then, the generation of the d sub-sequences of $\mathcal{S}$ 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 $\mathcal{S}$ is generated at frequency f.

The study of the γ parameter is out of the scope of this work 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.

3 Previous Results

The decimation of LFSR sequences has been used in cryptography in the design of new stream ciphers [MR84]. There exist 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. This construction 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 3.1 ([Zie59, Rue86])**.** *Let $\mathcal{S}$ be a sequence produced by an LFSR whose characteristic polynomial $Q(X)$ is irreducible in $\mathbb{F}_2$ of degree n. Let α be a root of $Q(X)$ and let T be the period of $Q(X)$. Let $\mathcal{S}_d^i$ be a sub-sequence resulting from the d-decimation of $\mathcal{S}$. Then, $\mathcal{S}_d^i$ can be generated by an LFSR with the following properties:*

- *The minimum polynomial of α^d in $\mathbb{F}_{2^n}$ 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 $n^\star$ of $Q^\star(X)$ is equal to the multiplicative order of $Q(X)$ in $\mathbb{Z}(T^\star)$.*

In practice, the characteristic polynomial $Q^\star(X)$ can be determined using the Berlekamp-Massey algorithm [Mas69]. 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 n^* must always be smaller or equal to n.

Construction using a multiple steps LFSR. This method was first proposed by Lempel and Eastman [LE71]. 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. Let the original LFSR of size n be build of the cells (a_i) for $0 \leq i \leq n$. All the cells (a_i), such that $i \bmod d = k$, are gathered to form a sub-shift register, where $0 \leq k \leq d-1$. This is the basic operation to transform a LFSR into a sub-sequences generator with a multiple steps solution. In the case of the Fibonacci setup, let A_t denote the whole state of the LFSR. Then, we apply the update function F at the states $A_t, A_{t+1}, \ldots, A_{t+d-1}$ to obtain the new feedback functions. In the case of the Galois setup, we have to define a new feedback function at each feedback position.

Comparison. In the following Table, we have summarized the two methods used to synthesize the sub-sequences generator. By $wt(Q(X))$, we mean the Hamming weight of the characteristic polynomial 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 [LE71] uses a constant number of memory cells for the synthesis of the sub-sequences generator.

Method	Memory cells	Logic Gates
LFSR synthesis [Mas69]	$d \times n^\star$	$d \times wt(Q^\star)$
Multiple steps LFSRs [LE71]	n	$d \times wt(Q)$

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 at frequency $\frac{\lambda f}{d}$ 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 and $C_d = \lambda C$ the capacity of LFSRs corresponding, respectively, to $\mathcal{S}$ and $\mathcal{S}_d^i$. 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 m-Sequences

FCSRs were introduced by Klapper and Goresky in [KG93]. 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. As for the LFSRs, there exists a Fibonacci and a Galois setup [GK02].

The contribution of our work is to apply the two methods used in the previous section on the case of ℓ-sequences.

Construction using FCSR synthesis. There exist algorithms based on Euclid's algorithm [ABN04] or on lattice approximation [GK97], which can determine the smallest FCSR to produce $\mathcal{S}_d^i$. These algorithms use the first k bits of $\mathcal{S}_d^i$ 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 $\mathcal{S}_d^i$, 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 $\mathcal{S}_d^i$ and $\mathcal{S}$.

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

Lemma 4.1. *Let $\mathcal{S} = (s_0, s_1, s_2, \ldots)$ be a periodic sequence with period T. For a given $d > 1$ and $0 \leq i \leq d-1$, let $\mathcal{S}_d^i$ be the decimated sequence with period $T^\star$. Then, it must hold:*

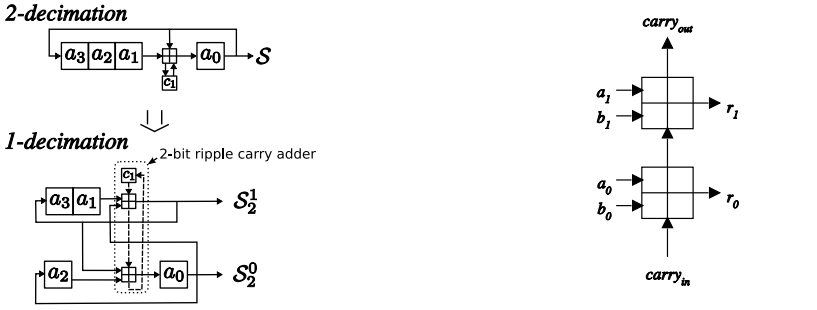
$$T^\star \left| \frac{T}{\gcd(T, d)} \right. . \tag{1}$$

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

In the case of $\gcd(T,d) > 1$, *the real value of* $T^\star$ *might depend on* i, *e.g.* for $\mathcal{S}$ being the 2-adic representation of $-1/19$ and $d = 3$ we have $T/gcd(T,d) = 6$, however, for $\mathcal{S}_3^0$ the period $T^\star = 2$ and for $\mathcal{S}_3^1$ 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 $gcd(T,d) = 1$, we know from [GK97] that $q^\star | 2^{T/2} + 1$. Based on a conjecture in [GK97] we can even assume that $q^\star$ is always bigger than q, if $gcd(T,d) = 1$. This means that the space complexity of this method is much worse than for the original FCSR which is also an interesting aspect for decimation attacks on FCSR sequences.

Construction using a multiple steps FCSR. We will apply the same technique than for the LFSR, however this time we have to take care of the carry path, *i.e.* we need the value of the carry at time t to compute the carry at time $t+1$. The computation of these subsequent carry bits seems to reduce the effectiveness of the decimation. However, it can be done efficiently by using n-bit ripple carry adders, which are well-known arithmetic circuits. An example of a multiple steps Galois FCSR can be found in Figure 2.



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

(b) 2-bit ripple carry adder.

Figure 2: Example for a Galois FCSR with $q = 19$.

References

[ABN04] François Arnault, Thierry P. Berger, and Abdelkader Necer. Feedback with carry shift registers synthesis with the euclidean algorithm. *IEEE Transactions on Information Theory*, 50(5):910–917, 2004.

[Fil00] Eric Filiol. Decimation attack of stream ciphers. In Bimal K. Roy and Eiji Okamoto, editors, *Progress in Cryptology - INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 31–42. Springer, 2000.

[FT91] Gui Liang Feng and Kenneth K. Tzeng. 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.

[GK97] Mark Goresky and Andrew Klapper. Arithmetic crosscorrelations of feedback with carry shift register sequences. *IEEE Transactions on Information Theory*, 43(4):1342–1345, 1997.

[GK02] Mark Goresky and Andrew Klapper. Fibonacci and Galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, 48(11):2826–2836, 2002.

[KG93] Andrew Klapper and Mark Goresky. 2-adic shift registers. In Ross J. Anderson, editor, *Fast Software Encryption - FSE'93*, volume 809 of *Lecture Notes in Computer Science*, pages 174–178. Springer, 1993.

[KG97] Andrew Klapper and Mark Goresky. Feedback shift registers, 2-adic span, and combiners with memory. *Journal of Cryptology*, 10(2):111–147, 1997.

[LE71] Abraham Lempel and Willard L. Eastman. High speed generation of maximal length sequences. *IEEE Transactions on Computers*, 2:227–229, 1971.

[LR08] Cédric Lauradoux and Andrea Röck. Parallel generation of ℓ-sequences. In Solomon W. Golomb, Matthew G. Parker, Alexander Pott, and Arne Winterhof, editors, *Sequences and Their Applications - SETA 2008*, volume 5203 of *Lecture Notes in Computer Science*, pages 299–312. Springer, 2008.

[Mas69] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inform. Theory*, 15:122–127, 1969.

[MR84] James L. Massey and Rainer A. Rueppel. Linear ciphers and random sequence generators with multiple clocks. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *Advances in Cryptology - EUROCRYPT'84*, volume 209 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 1984.

[Rue86] Rainer A. Rueppel. *Analysis and design of stream ciphers*. Springer-Verlag New York, Inc., New York, NY, USA, 1986.

[SC88] Ben J. M. Smeets and William G. Chambers. Windmill generators: A generalization and an observation of how many there are. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 325–330. Springer, 1988.

[SS06] Georg Schmidt and Vladimir Sidorenko. Linear shift-register synthesis for multiple sequences of varying length. In *IEEE International Symposium on Information Theory - ISIT 2006*, pages 1738–1742. IEEE, 2006.

[Zie59] Neal Zierler. Linear recurring Sequences. *Journal of the Society for Industrial and Applied Mathematics*, 2:31–48, 1959.