A Scalable Method for Constructing Galois NLFSRs with Period $2^n - 1$ using Cross-Join Pairs

Elena Dubrova

Royal Institute of Technology (KTH), Forum 120, 164 40 Kista, Sweden
{dubrova}@kth.se

Abstract. This paper presents a method for constructing n-stage Galois NLFSRs with period $2^n - 1$ from n-stage maximum length LFSRs. We introduce nonlinearity into state cycles by adding a nonlinear Boolean function to the feedback polynomial of the LFSR. Each assignment of variables for which this function evaluates to 1 acts as a crossing point for the LFSR state cycle. By adding a copy of the same function to a later stage of the register, we cancel the effect of nonlinearity and join the state cycles back. The presented method requires no extra time steps and it has a smaller area overhead compared to the previous approaches based on cross-join pairs. It is feasible for large n. However, it has a number of limitations. One is that the resulting NLFSRs can have at most $\lfloor n/2 \rfloor$-1 stages with a nonlinear update. Another is that feedback functions depend only on state variables which are updated linearly. The latter implies that sequences generated by the presented method can also be generated using a nonlinear filter generator.

1 Introduction

This paper addresses the problem of synthesis of Nonlinear Feedback Shift Registers (NLFSRs) with a guaranteed long period. This problem is known to be notoriously hard. None of the available algorithms is feasible for large NLFSRs.

One of the proposed approaches is to start from a shift register producing several shorter cycles and then to join them into one cycle. Pure cycling registers [10, 13, 14], pure summing registers [9], or LFSRs [20] has been used as a starting point in the previous algorithms.

An alternative approach is to start from a shift register with a known period and to obtain another shift register with the same period by using cross-join pairs [11]. The cross-join pair methods are based on the following idea. The new state cycle is obtained from a given state cycle C by first splitting C into two cycles, C_1 and C_2, and then by joining C_1 and C_2 back into another cycle. The splitting in done by interchanging the successors of a pair of states, $\{S_1, S_2\}$. This pair is called the *cross pair*. The joining in done by interchanging the successors of another pair of states, $\{S_3, S_4\}$, such that $S_3 \in C_1$ and $S_4 \in C_2$. This pair is called the *join pair*. Different methods use different strategies for selecting cross and join pairs [4, 12].

Helleseth and Kløve [18] proved that the number of cross-join pairs in a maximum length n-stage LFSR equals to $(2^{n-1}-1)(2^{n-1}-2)/6$. This important result provides a theoretical base for constructing n-stage NLFSRs with period $2^n - 1$ from maximum length LFSRs. Unfortunately, the problem of choosing a best set of compatible

cross-join pairs minimizing the circuit complexity of feedback functions of the resulting NLFSR is intractable.

In this paper, we show that the circuit complexity of feedback functions can be easily controlled if we use the Galois instead of the Fibonacci configuration of NLFSRs. We present a method in which the cross-join pairs are determined by a nonlinear function which is added to the input stage of the LFSR. Each assignment of variables for which this function evaluates to 1 acts as a crossing point for the original LFSR cycle. By adding a copy of the same nonlinear function to a later stage of the LFSR, we join the cycles back.

Since our method works with Boolean functions rather than with state cycles, it is feasible for large n. However, it has a number of limitations. One is that the resulting NLFSRs can have at most $\lfloor n/2 \rfloor$-1 stages with a nonlinear update. Another is that feedback functions depend only on state variables which are updated linearly. The latter implies that sequences generated by the presented method can also be generated using a nonlinear filter generator.

The paper is organized as follows. Section 2 gives a background on FSRs. Section 3 describes previous work. Section 4 presents our method for constructing NLFSRs. Section 5 discusses the relation between NLFSRs constructed using the presented approach and nonlinear filter generators. Section 6 concludes the paper.

2 Preliminaries

Throughout the paper, we use "$+$" and "$\cdot$" to denote addition and multiplication in $GF(2)$. We use $\bar{x}$ to denote the complement of x, which is defined by $\bar{x} = 1 + x$. The Boolean functions $GF(2^n) \to GF(2)$ are represented using the *Algebraic Normal Form (ANF)* [23] which is a polynomial in $GF(2)$ of type

$$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{2^n-1} c_i \cdot x_0^{i_0} \cdot x_1^{i_1} \cdot \ldots \cdot x_{n-1}^{i_{n-1}},$$

where $c_i \in \{0,1\}$ and $(i_0 i_1 \ldots i_{n-1})$ is the binary expansion of i.

A *Feedback Shift Register (FSR)* consists of n binary storage elements, called *stages*. Each stage $i \in \{0,1,\ldots,n-1\}$ has an associated *state variable* x_i which represents the current value of the stage i and a *feedback function* $f_i : GF(2^n) \to GF(2)$ which determines how the value of i is updated (see Figure 1). For any $i \in \{0,1,\ldots,n-1\}$, f_i depends on $x_{(i+1) \bmod n}$ and a subset of variables from the set $\{x_0, x_1, \ldots, x_i\}$.

A *state* of an FSR is a vector of values of its state variables $(x_0, x_1, \ldots, x_{n-1})$. At every clock cycle, the next state of an FSR is determined from its current state by simultaneously updating the value of each stage i to the value of f_i. The *period* of an FSR is the length of the longest cyclic output sequence it produces [16]. The *output* of an FSR is the value of its stage 0. The *input* of an FSR is the value of its stage $n-1$.

If all feedback functions of an FSR are linear, then it is called a *Linear Feedback Shift Register* (LFSR). Otherwise, it is called a *Non-Linear Feedback Shift Register* (NLFSR).

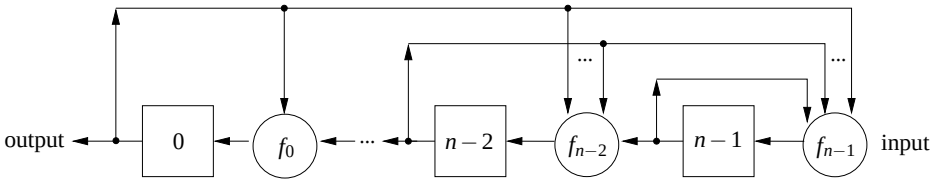

Fig. 1. The general structure of an FSR.

An FSR can be implemented either in the *Fibonacci* or in the *Galois* configuration In the former, the feedback is applied to the input stage of the shift register only, while in the latter the feedback can potentially be applied to every stage.

Its is known that the recurrence relation generated by the feedback function of an n-stage LFSR has a characteristic polynomial of degree n [16]. If this polynomial is primitive, then the state space of the LFSR consists of two cycles, of length 2^n-1 and 1. The cycle of length 2^n-1 includes all but all-0 state. The output sequences corresponding to this cycle are called *maximum length* sequences, or m-sequences.

Let $S=(s_0,s_1,\ldots,s_{n-1})$ be a state of an n-bit FSR. The *conjugate* $\hat{S}$ and the *companion* S' of S are defined by [10]:

$$\hat{S}=(\bar{s}_0,s_1,\ldots,s_{n-1}),$$
$$S'=(s_0,s_1,\ldots,\bar{s}_{n-1}).$$

In the Fibonacci configuration of FSRs, each state has two possible predecessors and two possible successors. The predecessors are conjugates of each others. The successors are companions of each others. The transitions between these four states define an *adjacency quadruple* associated with the binary $(n-1)$-tuple which they have in common (see Figure 2).

Two pairs of states $\{S_1,S_2\}$ and $\{S_3,S_4\}$ are called a *cross-join pair* if by interchanging the successors of S_1 and S_2 we split the state cycle into two cycles, and by interchanging the successors of S_3 and S_4 we join the two cycles back into one cycle [11]. In the traditional cross-join pair based methods, $S_2=\hat{S}_1$ and $S_4=\hat{S}_3$ [18]. In the method presented in this paper, $S_2=\hat{S}_1$, but $S_4\neq\hat{S}_3$.

An n-stage NLFSR is called *uniform* if, for some $0\leq\tau<n$:

$$f_i(x_0,x_1,\ldots,x_{n-1})=x_{(i+1)\bmod n}+h_i(x_0,\ldots,x_\tau)\ \text{ for }\ \tau\leq i<n$$
$$f_i(x_{i+1})=x_{i+1}\ \text{ for }\ 0\leq i<\tau$$

where h_i does not depend on the variable $x_{(i+1)\bmod n}$. The stage τ is called the *terminal stage* [8].

Let f_i and f_j be feedback functions of the stages i and j of an n-stage NLFSR, respectively. The operation *shifting*, denoted by $f_i\xrightarrow{M}f_j$, moves a set of ANF monomials M from f_i to f_j. The index of each variable x_k in each monomial in M is changed to $x_{(k-i+j)\bmod n}$.

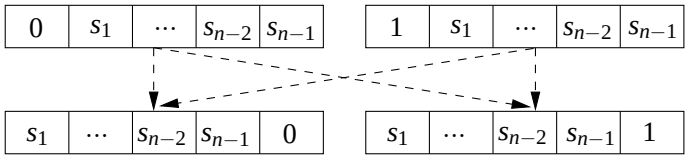

Fig. 2. An adjacency quadruple describing possible transitions between four states of an FSR in the Fibonacci configuration associated with the binary $(n-1)$-tuple $(s_1, s_2, \ldots, s_{n-1})$ which they have in common.

The following theorem, proved in [8], describes a sufficient condition for equivalence of NLFSRs before and after shifting. Two NLFSRs are *equivalent* if their sets of output sequences are equal.

Theorem 1. *Given a uniform NLFSR with the terminal stage* τ*, a shifting* $f_\tau \xrightarrow{M} f_{\tau'}$*,* $\tau' < \tau$*, results in an equivalent NLFSR if the transformed NLFSR is uniform as well.*

3 Previous work

Most of the previous works focused on the synthesis of full-length NLFSRs with period 2^n. Sequences generated by full-length NLFSRs are usually called *de Bruijn sequences* [7]. An excellent survey of algorithms for generating de Bruijn sequences given in [11].

Early algorithms used the *circulating register* [22] (also referred to as the *pure cycling register* [25]) as a starting point. A circulating register is an n-stage FSR whose feedback functions are of type $f_i(x_{(i+1) \bmod n}) = x_{(i+1) \bmod n}$ for $i \in \{0, 1, \ldots, n-1\}$. In other words, the content of the output stage of the register is cycled around to the input stage. This partitions the space of 2^n n-tuples into Z_n cycles, where

$$Z_n = \frac{1}{n} \sum_{d/n} \phi(d) 2^{n/d}$$

the sum is over all divisors d of the register length n and ϕ is the Euler's totient function [11].

The circulating register generates Z_n state cycles. It is known that, if the feedback function of the input stage of an FSR in the Fibonacci configuration is of type $f_{n-1}(x_0, x_1, \ldots, x_{n-1}) = x_0 + h(x_1, \ldots, x_{n-1})$, then the number of it state cycles changes by one as the weight of the truth table of h changes by one [11]. So, changing any of the 0s in the truth table of h either joins or splits two of the Z_n cycles. The cycles of the circulating register cannot be splitted since for this to happen a cycle of length n has to contain an even and an odd number of 1s. Therefore, with a single change from 0 to 1 two cycles can be joined together. This idea has been exploited to join cycles in different ways.

Circulating registers were studied by Golomb [16], Fredricksen [12] and Roth [29]. The number of possible interconnections between these cycles was derived by Mykkel-

veit [25] and van Lantschoot [22]. Fredricksen [10] have shown how to generate 2^{2n-5} full cycles from a circulating register of length n using $6n$ bits of storage and n time steps to produce the next state from a current state. Algorithms using composition to generate full cycles were developed by Fredricksen and Kessler [13], Etzion and Lempel [9], and a generalization to k-ary sequences was done by Fredricksen and Maiorana [14] and Ralston [26].

Etzion and Lempel [9] presented an algorithm for the generation of full cycles based on *pure summing registers*, whose feedback function of the input stage is of type $f_{n-1}(x_0, x_1, \ldots, x_{n-1}) = x_0 + x_1 + \ldots + x_{n-1}$. Their algorithm generates $2^{n^2/4}$ full cycles using $n^2/4$ bits of storage and n time steps to produce the next state from a current state.

Jensen [20] presented an algorithm for joining state cycles arising from an arbitrary shift register. This algorithm is based on the property that, if there exists a state S in a cycle C_1 and a state S' in a cycle C_2 which is a companion of S, then two cycles can be joined by interchanging the predecessors of S and S'. In its essence, the algorithm modifies the original feedback function of the shift register by complementing an entry in its truth table every time it encounters a special state, called the *state representative*. of the cycle. Therefore, the run-time of the algorithm is directly proportional to the time it takes to find cycle representatives. The shorter are the cycles of the original shift register, the faster is the algorithm. Jensen has shown that, if LFSR with short state cycles are used as a starting point, then such an approach makes possible to generate $O(2^{2n/log2n})$ full length sequences using $3n$ bits of storage and at most $4n$ time steps for producing the next state from a current state.

An algorithm based on cross-join pairs was proposed by Fredriksen [12] for a class of NLFSRs generating "prefer one" de Bruijn sequences. In a "prefer one" de Bruijn sequence, the n-tuple $(1, a_1, a_2, \ldots, a_{n-1})$ precedes the n-tuple $(0, a_1, a_2, \ldots, a_{n-1})$ for all $n-1$-tuples $(a_1, a_2, \ldots, a_{n-1})$ except all-0.

A recursive algorithm for generating de Bruijn sequences based on Lempels D-homomorphism was presented by Annexstein [1]. A more efficient, non-recursive algorithm based on Lempels D-homomorphism was given by Chang et al in [4]. In these algorithms, n-variable Boolean functions generating de Bruijn sequence of order n are constructed from Boolean functions with a smaller number of variables.

Janicka-Lipska and Stoklosa [19] presented a heuristic algorithm for random generation of feedback functions for full-length NLFSRs. The algorithm is based on the properties which are derived based on the results of an exhaustive search for feedback functions of full length NLFSRs of up to 6 variables. It is conjectured that these properties hold for functions with a larger number of variables.

Chang et al [5] conjectured that the number of cross-join pairs in maximum length LFSRs equals to $(2^{n-1}-1)(2^{n-1}-2)/6$. Helleseth and Kløve [18] gave an elegant proof of this conjecture.

4 Constructing NLFSRs from LFSRs

The following Theorem describes the basic step of the presented method for constructing n-stage Galois NLFSRs with period $2^n - 1$ from n-stage maximum length LFSRs.

Theorem 2. *Let N be an n-stage NLFSR with the feedback functions of type*

$$\begin{aligned} &f_{n-1}(x_0,x_1,\ldots,x_{n-2}) = x_0 + f_L(x_1,x_2,\ldots,x_{n-2}) + f_N(x_1,x_2,\ldots,x_{n-2}) \\ &f_{n-2}(x_0,x_1,\ldots,x_{n-3},x_{n-1}) = x_{n-1} + f_N(x_0,x_1,\ldots,x_{n-3}) \\ &f_{n-3}(x_{n-2}) = x_{n-2} \\ &\ldots \\ &f_0(x_1) = x_1 \end{aligned} \tag{1}$$

where f_L is a linear Boolean function of type

$$f_L(x_1,x_2,\ldots,x_{n-2}) = c_1x_1 + \ldots + c_{n-2}x_{n-2}$$

where $c_i \in \{0,1\}$, for $i \in \{1,2,\ldots,n-2\}$, and f_N is an arbitrary nonlinear Boolean function. Then N has period 2^n-1 if the polynomial

$$g(x) = 1 + c_1x + c_2x^2 + \ldots + c_{n-2}x^{n-2} + x^n$$

is primitive.

Proof: The primitive characteristic polynomial $g(x) = 1+c_1x+c_2x^2+\ldots+c_{n-2}x^{n-2}+x^n$ gives rise to an n-stage LFSR in the Fibonacci configuration whose input stage is updated using the function

$$f_{n-1}(x_0,x_1,\ldots,x_{n-2}) = x_0 + c_1x_1 + \ldots + c_{n-2}x_{n-2} = x_0 + f_L(x_1,x_2,\ldots,x_{n-2})$$

and $f_i(x_{i+1}) = x_{i+1}$, for $i \in \{0,1,\ldots,n-2\}$. Since $g(x)$ is primitive, the LFSR period is 2^n-1 [16].

Since $f+f=0$ for any Boolean function f, we can re-write the above as

$$f_{n-1}(x_0,\ldots,x_{n-2}) = x_0 + f_L(x_1,\ldots,x_{n-2}) + f_N(x_1,x_2,\ldots,x_{n-2}) + f_N(x_1,x_2,\ldots,x_{n-2})$$

where f_N is an arbitrary nonlinear Boolean function.

By applying the shifting $f_{n-1} \overset{M_{f_N}}{\rightarrow} f_{n-2}$, where M_{f_N} is the set of all monomials of the ANF of f_N, we obtain:

$$\begin{aligned} &f_{n-1}(x_0,x_1,\ldots,x_{n-2}) = x_0 + f_L(x_1,x_2,\ldots,x_{n-2}) + f_N(x_1,x_2,\ldots,x_{n-2}) \\ &f_{n-2}(x_0,x_1,\ldots,x_{n-3},x_{n-1}) = x_{n-1} + f_N(x_0,x_1,\ldots,x_{n-3}) \end{aligned}$$

and $f_i(x_{i+1}) = x_{i+1}$ for $i \in \{0,1,\ldots,n-3\}$. By Theorem 1, the output stage of this NLFSR generates the same sets of sequences as the output stage of the LFSR, therefore its period is 2^n-1.

□

Figure 3 illustrates the effect of nonlinear functions on the state cycle of the LFSR. Each non-0 assignment $(s_1,s_2,\ldots,s_{n-2}) \in GF(2^{n-2})$ of variables $(x_1,x_2,\ldots,x_{n-2})$ for which $f_N(x_1,x_2,\ldots,x_{n-2})$ evaluates to 1 complements two entries in the truth table of $f_{n-1}(x_0,x_1,\ldots,x_{n-2})$. Thus, the values of the input stages of the successors of the states $(0,s_1,\ldots,s_{n-2},s_{n-1})$ and $(1,s_1,\ldots,s_{n-2},s_{n-1})$ are complemented. For each $s_{n-1} \in \{0,1\}$, this splits the LFSR state cycle into two cycles.

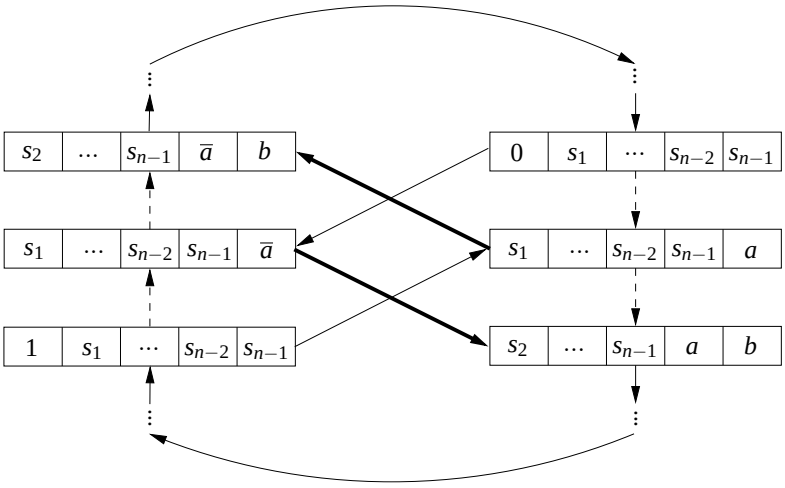

Fig. 3. An illustration of the effect of nonlinear functions on the state cycle of the LFSR for the case when $f_N(0,0,\ldots,0) = 0$. The dashed lines show the connections in the original LFSR state cycle. The solid lines show the connections in the constructed NLFSR state cycle; $a,b \in \{0,1\}$ are constants.

Since the function $f_N(x_0,x_1,\ldots,x_{n-3})$ which is added to $f_{n-2}(x_0,x_1,\ldots,x_{n-3},x_{n-1})$ evaluates to 1 for the assignment $(s_1,s_2,\ldots,s_{n-2})$ of variables $(x_0,x_1,\ldots,x_{n-3})$, two entries in the truth table of $f_{n-2}(x_0,x_1,\ldots,x_{n-3},x_{n-1})$ are also complemented. As a result, the values of the $n-2$nd stages of the successors of the states $(s_1,\ldots,s_{n-2},s_{n-1},a)$ and $(s_1,\ldots,s_{n-2},s_{n-1},\bar{a})$ are complemented. For each $s_{n-1} \in \{0,1\}$, this joints the previously splitted LFSR cycles back into one cycle.

We can see that these transformations result in interchanging pairs of companion states in the LFSR cycle. For each non-0 assignment $(s_1,s_2,\ldots,s_{n-2}) \in GF(2^{n-2})$, two pairs of companion states, $\{(s_1,\ldots,s_{n-2},0,a),(s_1,\ldots,s_{n-2},0,\bar{a})\}$ and $\{(s_1,\ldots,s_{n-2},1,a),(s_1,\ldots,s_{n-2},1,\bar{a})\}$, are interchanged.

Figure 4 illustrates the special case of all-0 assignment. If $f_N(0,0,\ldots,0) = 1$, then the all-0 state is joined into the main cycle and the state $(0,\ldots,0,1)$ is splitted from the main cycle. The states $(0,\ldots,0,1,0)$ and $(0,\ldots,0,1,1)$ are interchanged.

From the above, we can easily derive the following property. Let $wt(f)$ denote the Hamming weight of the truth table of a Boolean function f. Let $HD(a,b)$ denote the Hamming distance between the binary vectors a and b.

Lemma 1. *Let N be an n-stage NLFSR constructed using Theorem 2 and L be its underlying maximum length LFSR in the Fibonacci configuration. Let A_L and A_N be sequences of the input stages of the L and N, respectively, of length 2^n-1. If L and N are initialized to the same state which is neither $(0,\ldots,0,0)$ nor $(0,\ldots,0,1)$, then*

$$HD(A_L,A_N) = \begin{cases} 4 * wt(f_N), \text{ if } f_N(0,0,\ldots,0) = 0 \\ 4 * wt(f_N) - 1, \text{ otherwise.} \end{cases} \quad (2)$$

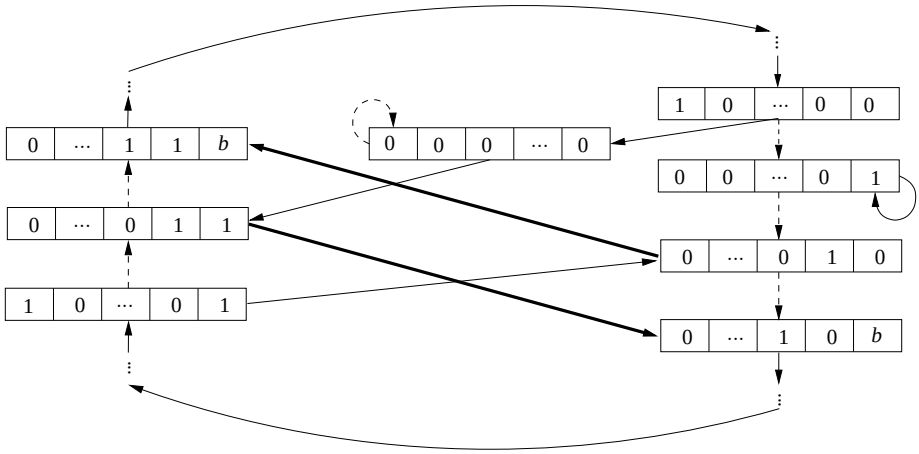

Fig. 4. The special case of $f_N(0,0,\ldots,0) = 1$.

Note, that it is important that in Theorem 2 neither f_L nor f_N depend on the variable x_{n-1}. Otherwise, the NLFSR after shifting $f_{n-1} \stackrel{M_{f_N}}{\rightarrow} f_{n-2}$ is not uniform and the equivalence to the original LFSR in not guaranteed by Theorem 1. As an example, consider the following 4-stage NLFSR in which f_N depends on the variable x_3:

$$
\begin{aligned}
f_3(x_0,x_1,x_2,x_3) &= x_0 + x_1 + x_1x_2x_3 \\
f_2(x_0,x_1,x_2,x_3) &= x_3 + x_0x_1x_2 \\
f_1(x_2) &= x_2 \\
f_0(x_1) &= x_1
\end{aligned}
$$

This NLFSR is constructed by adding the nonlinear function $x_1x_2x_3$ to the linear function $x_0 + x_1$ corresponding to the primitive polynomial $1 + x + x^4$. The resulting state cycles are shown in Figure 5. Their period is 1.

In general, it is not necessary to shift a copy of f_N to the feedback function of the $n-2$nd stage, as in Theorem 2. We can shift it to the stages with a lower index. We can also shift its monomials to several different feedback functions. The only requirement is to keep the NLFSR after shifting uniform, in order to preserve its equivalence to the NLFSR before shifting.

By shifting a copy of f_N to the stages with a lower index, we allow more stages of the resulting NLFSR to have a nonlinear update. For example, if the equations (1) are

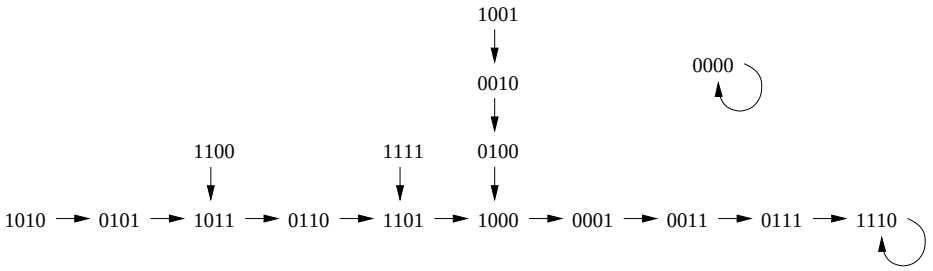

Fig. 5. An example of the state transition graph of an NLFSR constructed using Theorem 2 for the case when f_N depends on x_{n-1}.

modified to:

$$
\begin{aligned}
&f_{n-1}(x_0, x_{k-1}, x_k) = x_0 + f_L(x_1, x_2, \ldots, x_{n-k}) + f_N(x_{k-1}, x_k)\\
&f_{n-2}(x_{n-1}) = x_{n-1}\\
&\ldots\\
&f_{n-k+1}(x_{n-k+2}) = x_{n-k+2}\\
&f_{n-k}(x_0, x_1, x_{n-k+1}) = x_{n-k+1} + f_N(x_0, x_1)\\
&f_{n-k-1}(x_{n-k}) = x_{n-k}\\
&\ldots\\
&f_0(x_1) = x_1
\end{aligned}
$$

where $k = \lfloor n/2 \rfloor$, then the Theorem 2 still holds.

The above example shows that up to $\lfloor n/2 \rfloor - 1$ of the NLFSR constructed using the presented method can have a nonlinear update.

By shifting the monomials of f_N to several feedback functions, we can reduce the depth of feedback functions and share common monomials. This leads to faster and smaller NLFSRs [3]. For example, suppose that n is even and we use the nonlinear function $f_N(x_1, x_2, \ldots, x_2) = x_1x_2 + x_3x_4 + \ldots + x_{n-3}x_{n-2}$. Then, we can construct the following NLFSR with period $2^n - 1$:

$$
\begin{aligned}
&f_{n-1}(x_0, x_1, x_2) = x_0 + f_L(x_1, x_2) + x_1x_2\\
&f_{n-2}(x_0, x_1, x_{n-1}) = x_{n-1} + x_0x_1\\
&f_{n-3}(x_1, x_2, x_{n-2}) = x_{n-2} + x_1x_2\\
&f_{n-4}(x_0, x_1, x_{n-3}) = x_{n-3} + x_0x_1\\
&\ldots\\
&f_3(x_1, x_2, x_4) = x_4 + x_1x_2\\
&f_2(x_0, x_1, x_3) = x_3 + x_0x_1\\
&f_1(x_2) = x_2\\
&f_0(x_1) = x_1
\end{aligned}
$$

Such a distribution of monomials makes possible, in theory, to realize an n-stage NLFSR with period $2^n - 1$ using only 2 two-input AND gates and n two-input XOR gates. In practice, however, gates have a limited fan-out, i.e. their output can be connected only

to a limited number of gate inputs. Note, that the linear function f_L may depend on more variables, but then its variables with indexes larger that 2 have to be shifted down in order to get a uniform NLFSR.

By shifting the monomials of f_N to several feedback functions, we can also obtain a different type of nonlinear update for different stages of the NLFSR. For example, if $n = 6$, the primitive polynomial is $1+x+x^6$, and the nonlinear function is $f_N(x_1,x_2,x_3,x_4) = x_1x_2 + x_4 + x_3x_4$, then we can construct the following NLFSR:

$$\begin{aligned}
f_5(x_0,x_1,x_2) &= x_0 + x_1 + x_1x_2\\
f_4(x_0,x_1,x_5) &= x_5 + x_0x_1\\
f_3(x_1,x_2,x_4) &= x_4 + x_2 + x_1x_2\\
f_2(x_0,x_1,x_3) &= x_3 + x_1 + x_0x_1\\
f_1(x_2) &= x_2\\
f_0(x_1) &= x_1
\end{aligned}$$

5 Relation to Nonlinear Filter Generators

A known approach to improving the security of LFSRs is to pass the outputs of selected stages of an LFSR into a nonlinear filtering function which combines them to create a keystream. Such *nonlinear filter generators* are an important building block in many stream ciphers [27].

There is an obvious relation between sequences generated by NLFSRs constructed using the presented approach and nonlinear filter generators. A filter generator using $x_0 + f_L + f_N$ as its filtering function can produce the same sets of sequences as the input stage of an NLFSR constructed using the Theorem 2. Furthermore, since the NLFSR contains two copies of f_N, the filter generator will be smaller than the NLFSR.

However, if not only a sequence, but also its shifted versions are required, then NLFSRs will be more hardware-efficient than filter generators. We can use up to $\lfloor n/2 \rfloor - 1$ stages for shifting the content of the input stage. To obtain the same set of sequences with a nonlinear filter generator, an additional shift register is needed. Since flip-flops are more expensive than gates, this is less efficient than adding a second copy of f_N. For example, in UMC 90nm CMOS technology, the smallest flip-flop is 4.2 times larger than a two-input NAND gate.

Due to the relation between nonlinear filter generators and NLFSRs, the attacks on filter generators can be adopted to attack NLFSRs constructed using the presented approach (see Golić [15] and Rønjom, Gong and Helleseth [28] for excellent surveys of the various attacks on filter generators). On the other hand, we can make use of the accumulated knowledge on selecting Boolean functions for filter generators [6] and on linear complexity of their sequences [21, 24] to choose a cryptographically strong function f_N for the NLFSR.

Our results provide an alternative way of looking into sequences produced by nonlinear filter generators and might contribute to further understanding of their structure.

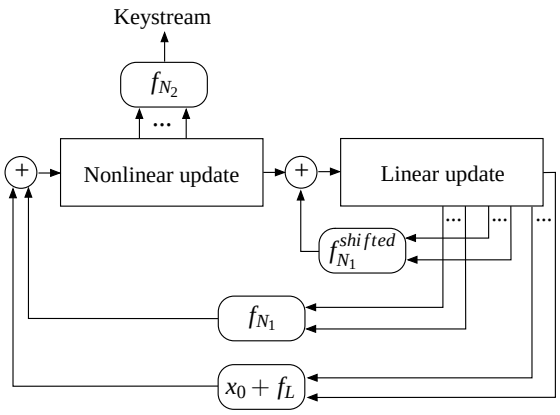

Fig. 6. A keystream generation method based on the presented NLFSRs.

6 Conclusion

In this paper, we presented a method for constructing n-stage NLFSRs in the Galois configuration with period $2^n - 1$ from maximum length LFSRs. Our method has no time overhead and it has a smaller area overhead compared to the previous approaches based on cross-join pairs. It is feasible for large n.

A limitation of the presented approach is that the resulting NLFSRs can have at most $\lfloor n/2 \rfloor$-1 stages with a nonlinear update. The rest of their stages is updated linearly. Another limitation is that feedback functions depend only on state variables which are updated linearly. Therefore the algebraic degree of the polynomial representing the value of the input stage of an NLFSR after t time steps will be at most by 1 larger than the algebraic degree of the polynomial representing the feedback function f_{n-1}.

The presented method might be useful for applications that require nonlinear sequences together with a set of their shifted versions, e.g. in the design of correlators for spread spectrum communication systems, randing systems, or radar systems. It will also be interesting to investigate if it can help us to construct a secure stream cipher which is smaller than the top stream ciphers such as Grain-128 [17] and Trivium [2]. Figure 6 shows a possible diagram in which the outputs of selected stages with the nonlinear update are passed to another nonlinear function, f_{N_2}, for creating a keystream.

References

1. F. S. Annexstein. Generating de Bruijn sequences: An efficient implementation. *IEEE Transactions on Computers*, 46:198 – 200, 1997.
2. C. Cannière and B. Preneel. Trivium. *New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986*, pages 244–266, 2008.
3. J.-M. Chabloz, S. Mansouri, and E. Dubrova. An algorithm for constructing a fastest Galois NLFSR generating a given sequence. In C. Carlet and A. Pott, editors, *Sequences and Their*

Applications - SETA 2010, volume 6338 of *Lecture Notes in Computer Science*, pages 41–54. Springer Berlin / Heidelberg, 2010.

4. T. Chang, B. Park, Y. H. Kim, and I. Song. An efficient implementation of the D-homomorphism for generation of de bruijn sequences. *IEEE Transactions on Information Theory*, 45:1280–1283, 1999.
5. T. Chang, I. Song, and S. H. Cho. Some properties of cross-join pairs in maximum length linear sequences. In *Proceeding of ISZTA '90*, pages 1077–1079, 1990.
6. T. W. Cusick and P. Stănică. *Cryptographic Boolean functions and applications*. Academic Press, San Diego, CA, USA, 2009.
7. N. G. de Bruijn. A combinatorial problem. *Nederl. Akad. Wetensch*, 49:758–746, 1946.
8. E. Dubrova. A transformation from the Fibonacci to the Galois NLFSRs. *IEEE Transactions on Information Theory*, 55(11):5263–5271, November 2009.
9. T. Etzion and A. Lempel. Algorithms for the generation of full-length shift register sequences. *IEEE Transactions on Information Theory*, 3:480–484, May 1984.
10. H. Fredricksen. A class of nonlinear deBruijn cycles. *J. Comb. Theory*, 19(A):192–199, Sept. 1975.
11. H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *SIAM Review*, 24(2):195–221, 1982.
12. H. M. Fredricksen. Disjoint cycles from de Bruijn graph. Technical Report 225, USCEE, 1968.
13. H. M. Fredricksen and I. J. Kessler. Lexicographic compositions and de Bruijn sequences. *J. Comb. Theory*, 22:17–30, 1977.
14. H. M. Fredricksen and J. Maiorana. Neklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Math.*, 23:207–210, 1978.
15. J. D. Golić. On the security of shift register based keystream generators. *Fast Software Encryption, LNCS 809*, pages 90–100, 1994.
16. S. Golomb. *Shift Register Sequences*. Aegean Park Press, 1982.
17. M. Hell, T. Johansson, A. Maximov, and W. Meier. The Grain family of stream ciphers. *New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986*, pages 179–190, 2008.
18. T. Helleseth and T. Kløve. The number of cross-join pairs in maximum length linear sequences. *IEEE Transactions on Information Theory*, 31:1731–1733, 1991.
19. I. Janicka-Lipska and J. Stoklosa. Boolean feedback functions for full-length nonlinear shift registers. *Telecommunications and Informatioin Technology*, 5:28–29, 2004.
20. C. J. Jansen. *Investigations On Nonlinear Streamcipher Systems: Construction and Evaluation Methods*. Ph.D. Thesis, Technical University of Delft, 1989.
21. E. Key. An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Transactions on Information Theory*, 22:732 – 736, 1976.
22. E. J. v. Lantschoot. Double adjacencies between cycles of a circulating shift register. *Transactions on Computers*, C-22:944–954, 1973.
23. R. Lidl and H. Niederreiter. *Introduction to Finite Fields and their Applications*. Cambridge Univ. Press, 1994.
24. J. L. Massey and S. Serconek. Linear complexity of periodic sequences: A general theory. In *Advances in Cryptology-Crypto'96, Lecture Notes in Computer Science*, pages 358–371. Springer-Verlag, 1996.
25. J. Mykkelveit. Generating and counting the double adjacencies in a pure cycling shift register. *Transactions on Computers*, C-24:299–304, 1975.
26. A. Ralston. A new memoryless algorithm for de Bruijn sequences. *J. Algorithms*, 2:50–62, 1981.
27. M. Robshaw. Stream ciphers. Technical Report TR - 701, July 1994.

28. S. Rønjom, G. Gong, and T. Helleseth. A survey of recent attacks on the filter generator. In S. Boztas and H.-F. Lu, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 4851 of *LNCS*, pages 7–17. Springer Berlin / Heidelberg, 2007.
29. E. Roth. Permitations arranged around a cycle. *Amer. Math. Monthly*, pages 990–992, 1971.