A Method for Generating Full Cycles by a Composition of NLFSRs

Elena Dubrova

the date of receipt and acceptance should be inserted later

Abstract Non-Linear Feedback Shift Registers (NLFSR) are a generalization of Linear Feedback Shift Registers (LFSRs) in which a current state is a non-linear function of the previous state. The interest in NLFSRs is mostly motivated by their ability to generate pseudo-random sequences which are usually hard to break with existing cryptanalytic methods. However, it is still not known how to construct large $n$-stage NLFSRs which generate full cycles of $2^n$ possible states. This paper presents a method for generating full cycles by a composition of NLFSRs. First, we show that an $n \times k$-stage register with period $O(2^{2n})$ can be constructed from $k$ $n$-stage NLFSRs by adding to their feedback functions a logic block of size $O(n \times k)$, for $k > 1$. This logic block implements Boolean functions representing the set of pairs of states whose successors have to be exchanged in order to join cycles. Then, we show how to join all cycles into one by using one more logic block of size $O(n \times k^2)$ and an extra time step.

Keywords LFSR · NLFSR · full cycle · de Bruijn sequence · composition

1 Introduction

Non-Linear Feedback Shift Registers (NLFSR) are a generalization of Linear Feedback Shift Registers (LFSRs) in which a current state is a non-linear function of the previous state [18]. The interest in NLFSRs is motivated to a large extent by their ability to generate pseudo-random sequences which are usually hard to break with existing cryptanalytic methods [23]. While LFSRs are widely used in testing and simulation [6], for cryptographic applications their pseudo-random sequences are not secure. The structure of an $n$-bit LFSR can be easily deduced from $2n$ consecutive bit of its sequence by using the Berlekamp-Massey algorithm [21]. Contrary, an adversary might need up to $O(2^n)$ bits of a sequence to determine the structure of the $n$-bit NLFSR which generates it [10].

E. Dubrova
Royal Institute of Technology (KTH), Forum 120, Kista, Sweden
Tel.: +46-8-7904114
Fax: +46-8-7511793
E-mail: dubrova@kth.se
However, while the theory behind LFSRs is well-understood, many fundamental problems related to NLFSRs remain open. One of the most important ones is constructing an NLFSR with the maximum period. It is known that an \( n \)-stage LFSR has the maximum period of \( 2^n - 1 \) if and only if its characteristic polynomial is primitive [18]. For NLFSRs, no similar property has been found so far. Small NLFSRs with the maximum period can be computed by simulation. However, with today’s processing power, we can simulate NLFSRs of size \( n < 35 \) only [7]. This is not enough for cryptographic applications, which require periods larger that \( 2^{128} \) [22].

Sequences generated by maximum-period NLFSRs are also known as de Bruijn sequences. In a de Bruijn sequence of order \( n \) all \( 2^n \) different binary \( n \)-tuples appear exactly once. It is known that the number of different de Bruijn sequences of order \( n \) is \( 2^{2^{n-1}-n} \) [3]. An excellent survey of algorithms for generating de Bruijn sequences using shift registers is given in [13]. These algorithms can be classified into two groups.

Algorithms in the first group start from a shift register producing several shorter cycles and then join them into one cycle. Fredricksen [12] have shown how to generate 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. The algorithm of Etzion and Lempel [11] generates full cycles from a pure summing register using \( n^2/4 \) bits of storage and \( n \) time steps to produce the next state. Jansen [20] presented an algorithm for joining state cycles of an arbitrary shift register. This algorithm generates full cycles using \( 3n \) bits of storage and at most \( 4n \) time steps for producing the next state from a current state.

A recursive algorithm for generating de Bruijn sequences based on Lempel’s D-homomorphism was presented by Annexstein [1]. A non-recursive version of this algorithm was proposed by Chang et al in [4]. In both algorithms, \( n \)-variable Boolean functions generating de Bruijn sequence of order \( n \) are constructed from Boolean functions with a smaller number of variables.

Algorithms in the second group start from a shift register with a known period and obtain another shift register with the same period by using cross-join pairs. Different approaches to selecting cross-join pairs have been explored. Fredricksen [14] proposed an algorithm 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. Dubrova [8] presented a method for constructing NLFSRs with period \( 2^n - 1 \) from maximum-period LFSRs in which the cross-join pairs are determined by a non-linear function added to the LFSR. The number of possible cross-join pairs in maximum-period LFSRs has been derived by Helleseth and Klove [19].

This paper presents a method for generating full cycles by a composition of NLFSRs. First, we show how to construct \( n+k \)-stage register with period \( O(2^{2k}) \) from \( k \) \( n \)-stage NLFSRs, for \( k > 1 \). We derive Boolean functions representing the set of pairs of states whose successors have to be exchanged in order to join cycles. These functions are added to the feedback functions of \( n \)-stage NLFSRs. We prove that the additional logic can be implemented using \( O(n+k) \) 2-input gates. Then, we show how to join all cycles into one by using one more logic block of size \( O(n+k^2) \) and an extra time step.

In a full cycle generated by a traditional NLFSR, each current and next state overlap in all but one positions. In \( n \times k \)-stage registers constructed using the presented method, each current and next state overlap in all but \( k \) positions. Therefore, their full cycles differ from the full cycles generated by NLFSRs and their output sequences are not of de Bruijn type.

The paper is organized as follows. Section 2 gives a background on NLFSRs. Section 3 introduces \((n, k)\)-composed registers. Section 4 analyses properties of \((n, k)\)-composed registers. Section 5 presents an algorithm for constructing registers with period \( O(2^{2k}) \) from
A Method for Generating Full Cycles by a Composition of NLFSRs

Section 6 presents an algorithm which joins all cycles into one. Section 7 concludes the paper.

2 Preliminaries

Throughout the paper, we use "\oplus" and "\cdot" to denote addition and multiplication in GF(2), respectively, and "+" and "\times" to denote arithmetic addition and multiplication, respectively. We use \( \pi \) to denote the complement of \( x \), defined by \( \pi = 1 \oplus x \).

An \( n \)-stage Non-Linear Feedback Shift Register (NLFSR) consists of \( n \) binary storage elements, called stages [18]. Each stage \( i \in \{1,2,\ldots,n\} \) has an associated state variable \( x_i \in \{0,1\} \) which represents the current value of the stage. At each clock cycle, the value of \( x_{i+1} \) is transferred to \( x_i \), for all \( i \in \{1,2,\ldots,n-1\} \). The feedback function \( f(x_1,x_2,\ldots,x_n) \), computed from the content of the \( n \) stages, determines the next value of \( x_n \). The output of an NLFSR is the sequence of bits appearing in its stage 1.

The feedback function \( f \) induces the mapping \( F : \{0,1\}^n \rightarrow \{0,1\}^n \) of type

\[
(x_1,x_2,\ldots,x_n) \rightarrow (x_2,x_3,\ldots,x_n,f(x_1,x_2,\ldots,x_n)).
\]

The state of an \( n \)-stage register is a vector of values \( S = (x_1,x_2,\ldots,x_n) \in \{0,1\}^n \) of its state variables \( x_1,x_2,\ldots,x_n \).

A cycle of length \( m \) of an \( n \)-stage register is a vector of states \( (S_0,S_1,\ldots,S_{m-1}) \) such that \( F(S_i) = S_{i+1} \), for \( i \in \{0,1,\ldots,m-2\} \), and \( F(S_{m-1}) = S_0 \). The period of a register is the length of its longest cycle.

A necessary and sufficient condition for an NLFSR to be branchless [18] is that its feedback function \( f \) can be written in the form

\[
f(x_1,x_2,\ldots,x_n) = x_1 \oplus g_1(x_2,\ldots,x_n),
\]

where \( g \) is a Boolean function which does not depend on the variable \( x_1 \).

A product-term of an \( n \)-variable Boolean function is an expression of type \( x_1 \cdot x_2 \cdot \ldots \cdot x_k \)
where \( x_i \) is either a variable \( x_i \) or its complement \( \pi \), for \( 1 \leq k \leq n \) [2]. A minterm of an \( n \)-variable Boolean function is a product-term with \( k = n \). A product-term represents a subspace of an \( n \)-dimensional Boolean space, while a minterm represents a point of an \( n \)-dimensional Boolean space.

3 Composition of NLFSRs

Consider an \( n \times k \)-stage register composed of \( k \) \( n \)-stage NLFSRs \( N_1,N_2,\ldots,N_k \) as shown in Figure 1. The combining function \( f \) is a Boolean function of type \( \{0,1\}^k \rightarrow \{0,1\} \). A lot of research has been done on characterizing Boolean functions which are cryptographically strong [5].

Note that the period of a shift register is traditionally defined as the length of the longest cyclic output sequence it produces [18]. For shift registers, both definitions are equivalent. However, for general registers, in which each stage can be updated by its own function, the length of the longest state cycle can be a multiple of the length of the longest cyclic output sequence. For example, the 2-stage register with the state cycle \( \langle (00),(11),(01),(10) \rangle \) of length 4 generates the cyclic output sequence \( (0,1) \) of length 2.
It is known [9] that the composition of two cycles $A = (S_{A,0}, S_{A,1}, \ldots, S_{A,|A|-1})$ of length $|A|$ and $B = (S_{B,0}, S_{B,1}, \ldots, S_{B,|B|-1})$ of length $|B|$ whose states depend on disjoint sets of variables result in a set of cycles

$$A \circ B = \bigcup_{i=0}^{d-1} D_i$$

where $d$ is the greatest common divisor of $|A|$ and $|B|$, each cycle $D_i$ is of length $m$, where $m$ is the least common multiple of $|A|$ and $|B|$, and the $j$th state of $D_i$ is a concatenation of $(j \mod |A|)$th state of $A$ and $((i + j) \mod |B|)$th state of $B$:

$$S_{D_i,j} = (S_{A,j \mod |A|}, S_{B,(j+p) \mod |B|})$$

for $i \in \{0, 1, \ldots, d-1\}$, $j \in \{0, 1, \ldots, m-1\}$.

For example, if we compose several NLFSRs whose periods are pairwise co-prime, then the resulting register has the period equal to the product of periods of the NLFSRs. Such a technique has been used to construct registers with a guaranteed long period for stream ciphers Achterbahn [15], VEST [17], and the cipher [16].

In this paper, we are focusing on the case when the NLFSRs $N_1, N_2, \ldots, N_k$ have the same size and period which, to our best knowledge, has not been considered.

**Definition 1** An $n \times k$-composed register is constructed from $k$ $n$-stage NLFSRs $N_i$, $i \in \{1, 2, \ldots, k\}$, $k > 1$, such that each $N_i$ has two cycles of the following type:

1. a cycle of length $2^n - 1$ which consists of all non-zero states of $N_i$, called non-zero cycle, and
2. a cycle of length one which consists of the all-zero state of $N_i$, called zero cycle.

The output of an $n \times k$-composed register is computed from the outputs of NLFSRs $N_1, N_2, \ldots, N_k$ using a Boolean function $f : \{0, 1\}^k \rightarrow \{0, 1\}$.

Note that, in a traditional NLFSR, each current and next state overlap in all but one positions. The $n \times k$-composed registers do not match this classical definition. Therefore, their full cycles differ from the full cycles generated by NLFSRs and their output sequences are not of de Bruijn type.

In the next section, we derive a number of important properties of $(n,k)$-composed registers.
4 Properties of \((n, k)\)-Composition Registers

Let \(x_{i,j} \in \{0, 1\}\) denote the state variable of the stage \(j\) of \(n\)-stage NLFSR \(N_i\), for \(i \in \{1, 2, \ldots, k\}\), \(j \in \{1, 2, \ldots, n\}\). Then a state of an \(n \times k\)-composed register \(N\) is an \(n \times k\)-tuple of type
\[
S = (s_{1,1}, s_{1,2}, \ldots, s_{1,n}, s_{2,1}, s_{2,2}, \ldots, s_{2,n}, \ldots, s_{k,1}, s_{k,2}, \ldots, s_{k,n}).
\]

The right complement of a state \(S\) of an \((n, k)\)-composed register is defined by
\[
S_R = (s_{1,1}, s_{1,2}, \ldots, s_{1,n}, s_{2,1}, s_{2,2}, \ldots, s_{2,n}, \ldots, s_{k,1}, s_{k,2}, \ldots, s_{k,n})
\]
and the left complement of a state \(S\) of an \((n, k)\)-composed register is defined by
\[
S_L = (\overline{s_{1,1}}, s_{1,2}, \ldots, \overline{s_{1,n}}, \overline{s_{2,1}}, s_{2,2}, \ldots, \overline{s_{2,n}}, \ldots, \overline{s_{k,1}}, s_{k,2}, \ldots, \overline{s_{k,n}}).
\]

For the case of \(k = 1\), the right complement of \(S\) is equivalent to the companion of \(S\) and the left complement of \(S\) is equivalent to the conjugate of \(S\) [14].

Let \(S^+\) denote the next state of \(S\). Then, the following property holds.

\[\text{Lemma 1} \quad \text{For every state } S \text{ of an } (n, k)\text{-composed register, } (S_L)^+ = (S^+)_R.\]

\[\text{Proof}\] For clarity, let us introduce an abbreviation \(s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n})\). Then, \(S\) is of type:
\[
S = (s_{1,1}, s_{1,2}, s_{2,1}, s_{2,2}, \ldots, s_{k,1}, s_{k})
\]
and the next state of \(S\) is of type:
\[
S^+ = (s_{1,1}, a_1, s_{1,2}, a_2, \ldots, s_{k,1}, a_k),
\]
for some constants \(a_1, a_2, \ldots, a_k \in \{0, 1\}\).

On the other hand, the left complement of \(S\) is of type:
\[
S_L = (\overline{s_{1,1}}, s_{1,2}, \overline{s_{2,1}}, s_{2,2}, \ldots, \overline{s_{k,1}}, s_{k})
\]
and the next state of \(S_L\) is of type:
\[
(S_L)^+ = (s_{1,1}, b_1, s_{1,2}, b_2, \ldots, s_{k,1}, b_k),
\]
for some constants \(b_1, b_2, \ldots, b_k \in \{0, 1\}\).

Since all NLFSRs \(N_i\) are branchless, for all \(i \in \{1, 2, \ldots, k\}\), their feedback functions \(f_i\) have the form
\[
f_i(x_{i,1}, x_{i,2}, \ldots, x_{i,n}) = x_{i,1} \oplus g_i(x_{i,2}, \ldots, x_{i,n}).
\]
Therefore, \(f_i(s_{1,1}, s_{1,2}, \ldots, s_{i,n}) \neq f_i(s_{i,1}, s_{i,2}, \ldots, s_{j,n}),\) for all \(i \in \{1, 2, \ldots, k\}\). So, we can conclude that \(b_i = s_{i,j}\) for all \(i \in \{1, 2, \ldots, k\}\). This implies that \((S_L)^+ = (S^+)_R\).

The decimal representation of a state \(S = (s_{1}, s_{2}, \ldots, s_{p})\) is defined as
\[
D(S) = \sum_{i=1}^{p} 2^{p-i} \cdot s_i.
\]

A state of a cycle with the minimal decimal representation is called the minimal state of the cycle, denoted by \(S_{\text{min}}\).

For the case of \(k = 1\), minimal state of the cycle is equivalent to the state representative of a cycle [20].

Next, we show that, in an \((n, k)\)-composed register, \(S_{\text{min}}\) and \((S_{\text{min}})_L\) always belong to different cycles.
**Theorem 1** For every cycle of an \((n, k)\)-composed register, \(S_{\text{min}}\) and \((S_{\text{min}})_L\) belong to different cycles.

**Proof** Consider \((n, k + 1)\)-composed register \(N'\) constructed from the NLFSRs \(N_1, N_2, \ldots, N_k, N_{k+1}\). Since \(N_1, N_2, \ldots, N_k\) induce an \((n, k)\)-composed register \(N\), we can partition the cycles of \(N'\) into two groups:

1. Cycles obtained by composing the zero cycle of \(N_{k+1}\) with all cycles of \(N\).
2. Cycles obtained by composing the non-zero cycle of \(N_{k+1}\) with all cycles of \(N\).

For the first group, the states of all cycles of \(N'\) are a concatenation of a state \(S\) of a cycle of \(N\) and the all-zero state of \(N_{k+1}\). Therefore, for each cycle in this group, its minimal state \(S'_{\text{min}}\) is a concatenation of the minimal state \(S_{\text{min}}\) of the corresponding cycle of \(N\) and the all-zero \(n\)-tuple:

\[
S'_{\text{min}} = (S_{\text{min}}, 0, 0, \ldots, 0).
\]

The left complement of \(S'_{\text{min}}\) is of type:

\[
(S'_{\text{min}})_L = ((S_{\text{min}})_L, 1, 0, \ldots, 0).
\]

Thus \((S'_{\text{min}})_L\) belongs to the second group of cycles of \(N'\).

For the second group, the minimal state \(S'_{\text{min}}\) of every cycle of \(N'\) is a concatenation of some state \(S\) of the corresponding cycle of \(N\) and the state of \(N_{k+1}\) with the decimal representation 1:

\[
S'_{\text{min}} = (S, 1, 0, \ldots, 0).
\]

Therefore, the left complement of \(S'_{\text{min}}\) is a concatenation of \(S_L\) and the all-zero \(n\)-tuple:

\[
(S'_{\text{min}})_L = (S_L, 0, 0, \ldots, 0).
\]

Thus \((S'_{\text{min}})_L\) belongs to the first group of cycles of \(N'\).

If \(S\) and its left complement \(S_L\) belong to different cycles then, as shown in Figure 2a, by exchanging \(S^+\) and \((S_L)^+\), we join these two cycles into one. Contrary, if \(S\) and \(S_L\) belong to the same cycle then, as shown in Figure 2b, by exchanging \(S^+\) and \((S_L)^+\), we split this cycle into two.

**Theorem 2** An \((n, k)\)-composed register has

\[
C_k = \sum_{i=0}^{k-1} 2^{i+n} + 1
\]

cycles, in which one cycle is a zero cycle and all other cycles are of length \(2^n - 1\).
Proof By induction of \( k \).

**Basic case:** Obviously, for the \( k = 1 \), \( C_1 = 2 \). We have one zero cycle and one non-zero cycle consisting of \( 2^n - 1 \) non-zero states.

**Induction step:** Suppose that the theorem holds for \( k \). Consider \((n, k+1)\)-composed register \( N' \) constructed from \( k+1 \) \( n \)-stage NLFSRs \( N_1, N_2, \ldots, N_k, N_{k+1} \). The NLFSRs \( N_1, N_2, \ldots, N_k \) induce an \((n, k)\)-composed register \( N \). By inductive hypothesis, \( N \) has one zero cycle and \( C_k - 1 \) other cycles of length \( 2^n - 1 \). In the proof, we refer to the latter cycles as **non-zero** cycles on \( N \).

We can partition the cycles of \( N' \) into four types:

1. A cycle obtained by composing the zero cycle of \( N \) with the zero cycle of \( N_{k+1} \). We get a single cycle of length one.
2. A cycle obtained by composing the zero cycle of \( N \) with the non-zero cycle of \( N_{k+1} \). We get a single cycle of length \( 2^n - 1 \).
3. Cycles obtained by composing the \( C_k - 1 \) non-zero cycles of \( N \) with the zero cycle of \( N_{k+1} \). We get \( C_k - 1 \) cycles of length \( 2^n - 1 \) each.
4. Cycles obtained by composing the \( C_k - 1 \) non-zero cycles of \( N \) with the non-zero cycle of \( N_{k+1} \). We get \((C_k - 1)\times(2^n - 1)\) cycles of length \( 2^n - 1 \) each.

Thus, the total number of non-zero cycles in \( N' \) is \( 1 + (C_k - 1)\times 2^n = \sum_{i=0}^{k} 2^{i^n} = C_{k+1} - 1 \).

As we can see from Theorem 2, the period of an \((n, k)\)-composed register is \( 2^n - 1 \). In the next section, we show how to increase this period by cycle joining.

5 **Constructing Registers with Period \( O(2^{2n}) \)**

In this section, we show how to obtain cycles with period \( O(2^{2n}) \) by adding to an \((n, k)\)-composed register using a logic of size \( O(n \times k) \).

5.1 Cycle joining transformations

First, we analyze cycles obtained by exchanging the states \( S_{\text{min}} \) and \( (S_{\text{min}})_L \) of all cycles of an \((n, k)\)-composed register.

**Theorem 3** If for every cycle of an \((n, k)\)-composed register, the successors of \( S_{\text{min}} \) and \( (S_{\text{min}})_L \), are exchanged, then the resulting register has \( C_{k-1} \) cycles of length \( (2^n - 1)\times 2^n \) and one cycle of length \( 2^m \).

**Proof** Consider \((n, k+1)\)-composed register \( N' \) constructed from the NLFSRs \( N_1, N_2, \ldots, N_k, N_{k+1} \). NLFSRs \( N_1, N_2, \ldots, N_k \) induce an \((n, k)\)-composed register \( N \) which, by Theorem 2, has one zero cycle and \( C_k - 1 \) other cycles of length \( 2^n - 1 \).

We partition the cycles of \( N' \) into the same four types of cycles as in the proof of Theorem 2:

1. For the cycle of type 1, its minimal state is all-zero. The left complement of the minimal state consists of \( k \) concatenated \( n \)-tuples \((100 \ldots 0)\).
2. For the cycle of type 2, its minimal state consists of \( k - 1 \) concatenated all-zero \( n \)-tuples followed by the \( n \)-tuple \((100 \ldots 0)\). The left complement of the minimal state consists of \( k - 1 \) concatenated \( n \)-tuples \((100 \ldots 0)\) followed by the all-zero \( n \)-tuple.
3. For each of the cycles of type 3, its minimal state is a concatenation of the minimal state of the corresponding cycle in $N$ with the all-zero $n$-tuple. The left complement of the minimal state is a concatenation of the of the left complement of the minimal state of the corresponding cycle in $N$ and the $n$-tuple $(100\ldots0)$.

4. For each of the cycles of type 4, its minimal state is a concatenation of some state of the corresponding cycle in $N$ with the $n$-tuple $(100\ldots0)$. The left complement of the minimal state is a concatenation of the of the left complement of some state of the corresponding cycle in $N$ and the all-zero $n$-tuple.

From the description above we can see that, for each cycle of type 4, the left complement of the minimal state of the cycle is either of type 3 or of type 1. It is of type 1 for the cycle whose minimal state consists of $k$ concatenated $n$-tuples $(100\ldots0)$. Since there are $(C_k-1)*(2^n-1)$ cycles of type 4, we can conclude that $(C_k-1)*(2^n-1)$ states in the cycles of type 3 are left complement of some minimal state of some cycle of type 4. The remaining one state in the cycles of type 3 consists of $k-1$ concatenated $n$-tuples $(100\ldots0)$ followed by the all-zero state. It is the left complement of the minimal state of the cycle of type 2.

It follows from the above that if, for every cycle of $N'$, by exchanging the next states of its $S_{\text{min}}$ and $(S_{\text{min}})_{\text{L}}$, we get $C_k$ cycles of length $(2^n-1)*2^n$ which have the “flower” structure shown in Figure 3. The middle part of each “flower” corresponds to a cycle of $N'$ of type 3. The “petal” parts are cycles of $N'$ of type 4 or 2 (one case). We also get one cycle of length $2^n$ which joins the all-zero state of $N'$ with the cycle of $N'$ whose minimal state consists of $k$ concatenated $n$-tuples $(100\ldots0)$.

Finally, we show that the cycles we described contain all states of $N'$. On one hand, from Figure 3 we can conclude that the overall number of states contained in $C_k-1$ “flower” cycles and the other cycles are $A = (C_k-1) *(2^n-1)*2^n$. On the other hand, if we sum up the number of states in the cycles of types 1, 2, 3 and 4, we get $1 + (2^n-1) + (C_k-1) *(2^n-1) + (2^n-1) *(2^n-1) = A$.

The functions $f(x_1,x_2,\ldots,x_n)$ and $f(x_1,x_2,\ldots,x_n) \oplus x_1x_2 \ldots x_n$, where $x_i$ is defined as:

$$x_i = \begin{cases} \overline{x}_i, & \text{if } s_i = 0 \\ x_i, & \text{if } s_i = 1 \end{cases}$$

evaluate to the same values for all assignments of their variables except $(s_1,s_2,\ldots,s_n)$. This implies that we can change the next state of a state $(x_1,x_2,\ldots,x_n)$ of an $n$-stage NLFSR with the feedback function $f(x_1,x_2,\ldots,x_n)$ from $(x_2,\ldots,x_n, f(x_1,x_2,\ldots,x_n))$ to $(x_2,\ldots,x_n, f(x_1,x_2,\ldots,x_n))$ by adding to $f$ the minterm $x_1x_2 \ldots x_n$. Consequently, an $(n,k)$-composed

![Fig. 3 The “flower” cycle structure.](image-url)
The set of minimal states of all cycles of an \((n,k)\)-composed register can be obtained by adding to the feedback function of every NLFSR \(N_i\) the minterms corresponding to the states \(S_{\text{min}}\) and \((S_{\text{min}})_L\).

5.2 Extra logic block

Next, we derive sum-of-product expressions representing the \(S_{\text{min}}\) and \((S_{\text{min}})_L\) of all cycles of an \((n,k)\)-composed register.

**Theorem 4** The set of minimal states of all cycles of an \((n,k)\)-composed register is represented by the following Boolean function:

\[
f_{\text{min}}(x_1, x_2, \ldots, x_{M_k}) = \bigvee_{i=0}^{k-1} (x_{i\cdot n + 1} \cdot \prod_{j=i\cdot n+2}^{n_k} \bar{x}_j) \lor \prod_{j=1}^{n_k} x_j \tag{1}
\]

and the set of left complements of its minimal states is represented by:

\[
f_{\text{min},L}(x_1, x_2, \ldots, x_{M_k}) = \bigvee_{i=0}^{k-1} (\prod_{j=i\cdot n+1}^{n} \bar{x}_j \cdot \prod_{p=0}^{k-1} (x_{n+p+i\cdot n + 2} \lor x_{n+p+1} \cdot \prod_{m=i\cdot n + p + 2}^{n-1} \bar{x}_m)) \lor \prod_{j=1}^{n-1} x_j \tag{2}
\]

where "\(\lor\)" stands for the Boolean OR.

**Proof** By induction of \(k\).

**Basic case:** \(k = 1\), i.e. \(N = N_1\). Then the equation (1) reduces to

\[
f_{\text{min}}(x_1, x_2, \ldots, x_n) = (x_1 \cdot \prod_{j=2}^{n} x_j) \lor \prod_{j=1}^{n} x_j
\]

which correspond to the minimal state \((1,0,\ldots,0)\) of the cycle of length \(2^n - 1\) of \(N_1\) and to the minimal state \((0,0,\ldots,0)\) of the zero cycle of \(N_1\).

The equation (2) reduces to

\[
f_{\text{min},L}(x_1, x_2, \ldots, x_n) = \prod_{j=1}^{n} x_j \lor (x_1 \cdot \prod_{m=2}^{n-1} \bar{x}_m)
\]

which correspond to the left complements of the minimal states of the cycle of length \(2^n - 1\) of \(N_1\) and zero cycle of \(N_1\), respectively.

**Induction step:** Suppose that the theorem holds for \(k\). Consider \((n,k+1)\)-composed register \(N\) constructed from \((n,k+1)\)-composed register \(N_1, N_2, \ldots, N_k\). The NLFSRs \(N_1, N_2, \ldots, N_k\) induce an \((n,k)\)-composed register \(N\). By inductive hypothesis, their set of minimal states is represented by the equation (1). The product-terms of the equation (1) are listed in the following table with \(k+1\) rows:

<table>
<thead>
<tr>
<th>(S_{\text{min}})</th>
<th>Corresponding product-term</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 ... k</td>
<td>(x_1 \cdot x_2 \cdot \ldots \cdot x_{M_k})</td>
</tr>
<tr>
<td>0 0 0 ... 0</td>
<td>(x_1 \cdot x_2 \cdot \ldots \cdot x_{M_k})</td>
</tr>
<tr>
<td>1 0 0 ... 0</td>
<td>(x_1 \cdot x_2 \cdot \ldots \cdot x_{M_k})</td>
</tr>
<tr>
<td>x 1 0 ... 0</td>
<td>(x_{n+1} \cdot x_{n+2} \cdot \ldots \cdot x_{M_k})</td>
</tr>
<tr>
<td>x x 1 ... 0</td>
<td>(x_{2n+1} \cdot x_{2n+2} \cdot \ldots \cdot x_{M_k})</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>x x x ... 1</td>
<td>(x_{(k-1)n+1} \cdot x_{(k-1)n+2} \cdot \ldots \cdot x_{M_k})</td>
</tr>
</tbody>
</table>
Cycles obtained by composing the non-zero cycle of \(S_{\text{min}}\). Corresponding product-term:

<table>
<thead>
<tr>
<th>((S_{\text{min}})_L)</th>
<th>(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 ... 1</td>
<td>(x_1\overline{x}_2 \overline{x}<em>3 \overline{x}<em>4 + \ldots + \overline{x}<em>m + x</em>{(k-1)m+1} \overline{x}</em>{(k-1)m+2} + \ldots + \overline{x}</em>{nk})</td>
</tr>
<tr>
<td>0 1 1 ... 1</td>
<td>(x_1 \overline{x}_2 \overline{x}<em>3 \overline{x}<em>4 + \ldots + \overline{x}<em>m + x</em>{(k-1)m+1} \overline{x}</em>{(k-1)m+2} + \ldots + \overline{x}</em>{nk})</td>
</tr>
<tr>
<td>x 0 1 ... 1</td>
<td>(x_{kr+1} \overline{x}<em>{kr+2} \overline{x}</em>{kr+3} + \ldots + \overline{x}<em>{kn} + x</em>{(k-1)kr+1} \overline{x}<em>{(k-1)kr+2} + \ldots + \overline{x}</em>{nk})</td>
</tr>
<tr>
<td>x x 0 ... 1</td>
<td>(x_{kr+1} \overline{x}<em>{kr+2} + \overline{x}</em>{kr+3} x_{kr+4} + \ldots + \overline{x}<em>{kn} + x</em>{(k-1)kr+1} \overline{x}<em>{(k-1)kr+2} + \ldots + \overline{x}</em>{nk})</td>
</tr>
<tr>
<td>x x x ... 0</td>
<td>(x_{(k-1)kr+1} \overline{x}<em>{(k-1)kr+2} + \ldots + \overline{x}</em>{nk})</td>
</tr>
</tbody>
</table>

In this table, \(0\) denotes an \(n\)-tuple consisting of all zeros: \(0 = (0,0,\ldots,0)\), \(1\) denotes an \(n\)-tuple consisting of 1 followed by \(n-1\) zeros: \(1 = (1,0,\ldots,0)\), and \(x\) denotes an \(n\)-tuple consisting of all \(x: x = (x,x,\ldots,x)\), where the symbol "\(x\)" means either 0 or 1.

As in the proof of Theorem 1, we partition the cycles of \((n,k+1)\)-composed register \(N'\) into two groups:

1. Cycles obtained by composing the zero cycle of \(N_{k+1}\) with all cycles of \(N\).
2. Cycles obtained by composing the non-zero cycle of \(N_{k+1}\) with all cycles of \(N\).

For each cycle in the first group, its minimal state \(S_{\text{min}}\) is a concatenation of the minimal state \(S_{\text{min}}\) of the corresponding cycle of \(N\) and the all-zero \(n\)-tuple, i.e. it is of type \(S_{\text{min}} = (S_{\text{min}}, 0)\). Therefore, the Boolean function \(f_{\text{min}}^1\) representing the set of minimal states of all cycles in the first group is of type

\[ f_{\text{min}}^1 = \bigvee_{i=1}^{k+1} p_i \cdot \overline{x}_{kr+1} x_{kr+2} \ldots x_{kn} \overline{x}_{(k-1)kr+1} \overline{x}_{(k-1)kr+2} + \ldots + \overline{x}_{nk} \]

where \(p_i\) is the product-term from the \(i\)th row of the table above.

For each cycle in the second group, its minimal state \(S'_{\text{min}}\) is a concatenation of some state \(S\) of the corresponding cycle of \(N\) and the state of \(N_{k+1}\) with the decimal representation \(1\), i.e. it is of type \(S_{\text{min}} = (S, 1)\). Since cycles an \((n+1,k)\)-composed register \(N'\) represent all possible combinations which can be obtained from the cycles of \(N_1, N_2, \ldots, N_n\), the set of minimal states in the second group is composed of all possible states of \((n,k)\)-composed register \(N\) and 1. Therefore, the Boolean function \(f_{\text{min}}^2\) representing the set of minimal states of all cycles in the second group is of type

\[ f_{\text{min}}^2 = x_{kr+1} \overline{x}_{kr+2} + \overline{x}_{kr+3} x_{kr+4} + \ldots + \overline{x}_{kn} + x_{(k-1)kr+1} \overline{x}_{(k-1)kr+2} + \ldots + \overline{x}_{nk} \]

By taking the Boolean OR of \(f_{\text{min}}^1\) and \(f_{\text{min}}^2\), we get equation (1) for \(k = k + 1\).

Since the left complement of an \(n\)-tuple \(0\) is \(1\) and vice versa, for the left complements of the minimal states of \((n,k)\)-composed register \(N\) we get the product-terms listed in Table 1.

For the cycles in the first group, the set of left complements of the minimal states \((n+1,k)\)-composed register \(N'\) consists of states \((S'_{\text{min}})_L = ((S_{\text{min}})_L, 1)\), for all left complements of the minimal states \((S_{\text{min}})_L\) of \(N\). Therefore, the Boolean function \(f_{\text{min}}^1\) representing the set of left complements of the minimal states of all cycles in the first group is of type

\[ f_{\text{min}}^1 = \bigvee_{i=1}^{k+1} p_i \cdot x_{kr+1} \overline{x}_{kr+2} + \ldots + \overline{x}_{kn} \overline{x}_{(k-1)kr+1} \overline{x}_{(k-1)kr+2} + \ldots + \overline{x}_{nk} \]

where \(p_i\) is the product-term from the \(i\)th row of the table above.
For the cycles in the second group, the set of left complements consists of states \((S'_{\text{min}})_L = (S_L, 0)\), where \(S_L\) a left complement of a state \(S\) of \(N\), for all possible states \(S\) of \(N\). Therefore, the Boolean function \(f'_{\text{min}}\). representing the set of left complements of the minimal states of all cycles in the second group is of type

\[
\tilde{f}_{\text{min}}^2 = \overline{X_{k+1}} \overline{X_{k+2}} \ldots \overline{X_{n(k+1)}}.
\]

By taking the Boolean OR of \(f'_{\text{min}}^1\) and \(f'_{\text{min}}^2\), we get equation (2) for \(k = k + 1\).

Some optimizations can be done to reduce the size of the expression \(f_{\text{min}} \lor f_{\text{min}}\). First, the product-terms in the first two rows of both tables in the proof of Theorem 3 can be joined with one cycle. In this section, we show how to join cycles of a register constructed using Theorem 3 into one cycle.

The general structure of a register with period \((2^n - 1) \ast 2^k\) constructed using Theorem 3 is shown in Figure 4.

6 Constructing Registers with the Maximum Period

In this section, we show how to join cycles of a register constructed using Theorem 3 into one cycle.

6.1 Cycle joining transformations

The approach described in the previous section allows us to construct \(n \ast k\)-stage registers which have \(\sum_{i=0}^{k} 2^{n+i}\) cycles of length \((2^n - 1) \ast 2^k\) and one cycle of length \(2^n\). The following Theorem shows how to join these cycles into one.

**Theorem 5** Let \(N\) be an \(n \ast k\)-stage register constructed using Theorem 3. If the following transformations are applied to each current state \(S\) of \(N\) before computing the next state:

\[T1:\] If \(D(S) \in \{2^{2^{m+1}} + 1, \ldots, 2^{2^{m+1}} - 2\}\), for \(i \in \{1, 2, \ldots, k - 2\}\), then \(S\) is transformed to the state \(S'\) such that \(D(S') = D(S) + 1\).

\[T2:\] If \(D(S) = 2^{2^{m+1}} - 1\) for \(i \in \{0, 1, \ldots, k - 3\}\), then \(S\) is transformed to the state \(S'\) such that \(D(S') = 2^{k \times 2^{m+1}}\).

\[T3:\] If \(D(S) = 2^{(k-2) \times 2^n} + 1\), then \(S\) is transformed to the state \(S'\) such that \(D(S') = 0\).

\[T4:\] If \(D(S) = 0\), then \(S\) is transformed to the state \(S'\) such that \(D(S') = D(S) + 1\),

then the resulting register has period \(2^{n+k}\).

\(^2\) A 2-input gate implements a binary Boolean operation of type \(\{0, 1\}^2 \rightarrow \{0, 1\}\).
Consider an $n \times k$-stage register $N$ constructed using Theorem 3. $N$ has $\sum_{i=0}^{k-2} 2^{i+n}$ cycles of length $(2^n - 1) \times 2^n$ and one cycle of length $2^n$. The cycles of length $(2^n - 1) \times 2^n$ have the "flower" structure shown in Figure 3.

We can connect all cycles of $N$ together by transforming the minimal state of the middle cycle of each "flower" into the minimal state of the middle cycle of the next "flower" before computing the next state of $N$, and finally appending to the resulting chain of "flowers" the cycle of length $2^n$, as shown in Figure 5.

From the proofs of Theorems 3 and 4, it follows that the set of minimal states of the middle cycles of "flowers" is given by

$$S_{\min} = \begin{array}{ccccccc}
1 & 2 & 3 & \ldots & k-1 & k \\
1 & 0 & 0 & \ldots & 0 & 0 \\
x & 1 & 0 & \ldots & 0 & 0 \\
x & x & 1 & \ldots & 0 & 0 \\
\vdots & & & & & & \\
x & x & x & \ldots & 1 & 0
\end{array}$$

The table above contains $k-1$ rows. For each $i \in \{0, 1, \ldots, k-2\}$, the row $i$ represents a block of $2^{i+n}$ minimal states of the middle cycle of a "flower" with the decimal representations $\{2^{i+n}, 2^{i+n} + 1, \ldots, 2^{i+n} + 2^n - 2\}$. We order the states in each block according to their $D(S)$. Then, for $i = \{1, 2, \ldots, k-2\}$, we can visit all states in a block by incrementing by 1 $D(S)$ of each state $S$ in the block but the last. This is done by the transformation $T1$. To visit all blocks, for $i \in \{0, 1, \ldots, k-3\}$, we go from the last state of the block $i$, which has $D(S) = 2^{i+n} + 1$, to the first state of the block $i + 1$, which has $D(S') = 2^{(i+1)+n}$. This is done by the transformation $T2$.

In order to append the cycle of length $2^n$ to the resulting chain of "flowers", we go from the last state of the last block, which has $D(S) = 2^{(k-2)+n} + 1$, to the minimal state of the cycle of length $2^n$, which is the all-0 state. This is done by the transformation $T3$. Finally, we close the cycle by the going from the all-zero state to first state of the first block. This is done by incrementing $D(S)$ of all-0 state by 1, i.e. by the transformation $T4$.

Thus, by applying the transformation's $T1, T2, T3$ and $T4$, we join all cycles of $N$ into one.
As an example, consider the simple case of $k = 2$ for which we only need to transform
the state $\langle 1, 0 \rangle$ into $\langle 0, 0 \rangle$ and vice versa. This can be done using an $n \times k - 1$ input NOR
 gate with inputs from all the stages but $(1, 1)$ and a 2-input XOR gate with inputs from
the stage $(1, 1)$ and the NOR gate and the output to the stage $(1, 1)$. In this way, we comple-
ment the value of the stage $(1, 1)$ only if all the remaining stages have the value 0. Since
$S_1 + S_3 + \ldots + S_{n+1} = S_1S_3\ldots S_{n+k}$ and the product-term $S_1S_3\ldots S_{n+k}$ is used for implementing
the function $f_{\min}$, it brings no extra cost.

In the next section, we show that, in the general case, the transformations $T_1$, $T_2$, $T_3$ and
$T_4$ require $O(n \times k^2)$ 2-input gates to be implemented.

The registers constructed using Theorem 5 require two time steps to compute the next
step from a current step. At the first step, we update the current state by applying the transforma-
tions $T_1$, $T_2$, $T_3$ and $T_4$. At the second step, we compute the next state from the resulting
updated state.

6.2 Extra logic block

In the general case of $k \geq 2$, we can implement the transformations $T_1$ and $T_4$ using an
$(k-2) \times n$-bit adder in which one of the operands is 0 or 1, supplied by a controlling OR
described below, and another operand is the content of the stages of the first $k-2$ n-bit
NLFSRs. The result of addition is fed back into the stages of the first $k-2$ n-bit NLFSRs.
The adder is controlled by an OR-gate with inputs from the product-terms implementing
the set states with $D(S) \in \{2^n, 2^n+1, \ldots, 2^{n+1} - 2\}$, for $i = \{1, 2, \ldots, k-2\}$, and the
all-zero state. The addition of 1 is performed only if one of the product-terms evaluates to
1. A $(k-2) \times n$-bit adder can be implemented with $O(n \times k)$ 2-input gates. The controlling
logic can also be implemented with $O(n \times k)$ 2-input gates.

The transformations $T_2$ and $T_3$ can be implemented by assigning to each stage $(i, j)$, for
every $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, k-2\}$, and for $i = 1$ and $j = k-1$, a 2-input XOR
gate, with inputs from the stage $(i, j)$ and a controlling OR gate described below and the output
to the stage $(i, j)$. The OR-gates take inputs from the product-terms implementing
the set of states with $D(S) = 2^{n+1} - 1$ for $i = \{0, 1, \ldots, k-2\}$. Only if one of the product-
terms evaluates to 1, the outputs of ORs fed by this product-term become 1. That results
in complementing values of all stages controlled by these ORs. We need $O(n \times k)$ control-
ing ORs and each OR can have up to $k-1$ inputs. Therefore, the controlling logic for the transforma-
tions $T_2$ and $T_3$ requires $O(n \times k^2)$ 2-input gates.

So, the overall complexity of implementing the transformations $T_1$, $T_2$, $T_3$ and $T_4$ is
$O(n \times k^2)$ 2-input gates. Note that many of the product-terms used for implementing $T_1$, $T_2$, $T_3$ and $T_4$ are also used for implementing $f_{\min}$ and $f_{\max}$, so they can be shared.
7 Conclusion

In this paper, we presented a method for constructing \(n \times k\)-stage registers with period \(2^{n+k}\) by a composition of \(k\ n\)-stage NLFSRs. First, we show that an \(n \times k\)-stage register with period \(O(2^{n+k})\) can be constructed from \(k\ n\)-stage NLFSRs by adding to their feedback functions a logic block of size \(O(n \times k)\). Second, we show how to join all cycles into one by using one more logic block of size \(O(n \times k^2)\) and an extra time step.

Future work involves security analysis of the presented method.

Acknowledgements This work was supported in part by the project No 621-2010-4388 from Swedish Research Council.

References