Feedback Shift Registers, 2-Adic Span, and Combiners With Memory

Andrew Klapper* Mark Goresky †

January 15, 2003

Abstract

Feedback shift registers with carry operation (FCSR's) are described, implemented, and analyzed with respect to memory requirements, initial loading, period, and distributional properties of their output sequences. Many parallels with the theory of linear feedback shift registers (LFSR's) are presented, including a synthesis algorithm (analogous to the Berlekamp-Massey algorithm for LFSR's) which, for any pseudorandom sequence, constructs the smallest FCSR which will generate the sequence. These techniques are used to attack the summation cipher. This analysis gives a unified approach to the study of pseudorandom sequences, arithmetic codes, combiners with memory, and the Marsaglia-Zaman random number generator. Possible variations on the FCSR architecture are indicated at the end.

Index Terms – **Binary sequence, shift register, stream cipher, combiner with memory, cryptanalysis, 2-adic numbers, arithmetic code, $1/q$ sequence, linear span.**

1 Introduction

Pseudorandom sequences, with a variety of statistical properties (such as high linear span, low autocorrelation and pairwise cross-correlation values, and high pairwise hamming distance) are important in many areas of communications and computing (such as cryptography, spread

*763H Anderson Hall, Dept. of Computer Science, University of Kentucky, Lexington, KY, 40506-0047, klapper@cs.uky.edu. Project sponsored by the Natural Sciences and Engineering Researh Council under Operating Grant OGP0121648 and the National Security Agency under Grant Number MDA904-91-H-0012. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notation hereon.

†School of Mathematics, Institute for Advanced Study, Princeton N. J. Partially supported by the Ellentuck Fund and National Science Foundation grant # DMS 9304580.

spectrum communications, error correcting codes, and Monte Carlo integration). Binary pseudorandom periodic sequences are often generated in hardware, using linear feedback shift registers (LFSRs), and are referred to as "linear recurrence sequences". Certain register cells are "tapped", their contents are added modulo 2, and the sum is returned to the first cell of the shift register. (See Figure 1. The $q_i \in \{0, 1\}$ indicate existence or nonexistence of a tap. The symbol $\oplus$ refers to addition modulo 2.)

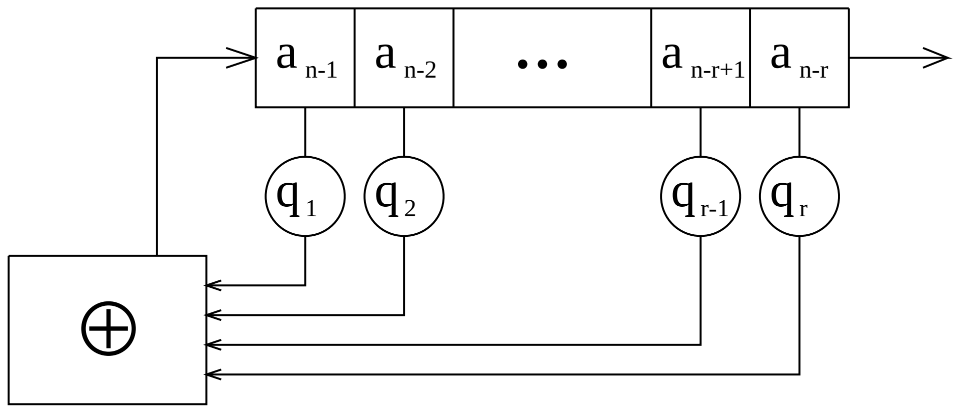

Figure 1: Linear Feedback Shift Register

Many modern stream ciphers are designed by combining the output of several LFSR's in various nonlinear ways (and hence ultimately depend on a deep understanding of the linear feedback architecture). Since 1955, an enormous amount of effort has been directed towards the study of other ("nonlinear") feedback architectures, (for example, [13], [44], [6]) which would give rise to fundamentally new or different kinds of pseudorandom sequences. However these other architectures have proven extremely resistant to analysis: even simple properties such as the period of nonlinear feedback shift register sequences are essentially unknown. Despite forty years of research, it is still the case that only the linear feedback architecture is adequately understood.

In this paper we introduce a new but very simple feedback architecture for shift register generation of pseudorandom sequences, which we call *"feedback with carry"* shift registers (FCSR). These are the shift register analogues of the Marsaglia-Zaman pseudorandom number generators [36]. It turns out that sequences generated by an FCSR share many of the important properties enjoyed by linear recurrence sequences. However, the analysis of FCSR sequences involves a completely different mathematical toolkit: instead of arithmetic in finite fields, we use arithmetic in the 2-adic numbers. Although some of the results in this paper may be proven without the use of the 2-adic numbers, we have been unable to find any alternate approach to the main results (Theorems 4.1, 9.5, 10.1, and 10.2).

An FCSR is a feedback shift register together with a small amount of auxiliary memory. In its simplest form, the cells in the register consist of bits (0 or 1), while the memory contains a nonnegative integer (see Figure 2).

The contents (0 or 1) of the tapped cells of the shift register are added *as integers* to the

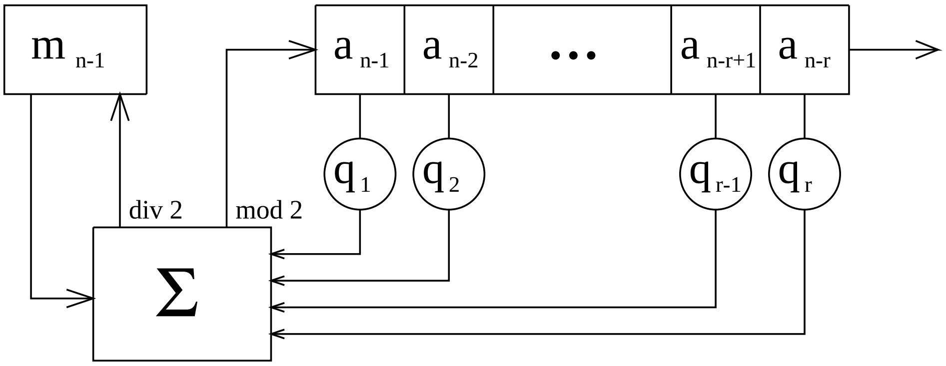

Figure 2: Feedback with Carry Shift Register

current contents of the memory to form a sum, σ. The parity bit ($\sigma \pmod 2$) of σ is fed back into the first cell, and the higher order bits ($\lfloor \sigma/2 \rfloor$) are retained for the new value of the memory. (More details of the feedback procedure and simple hardware implementation are described in Section 3).

The results in this paper are best described by exploiting the many parallels between properties of LFSR sequences and those of FCSR sequences. Let us first recall the salient features of the LFSR theory. Throughout this paper, $\mathbf{Z}$ denotes the integers, $\lfloor x \rfloor$ denotes the greatest integer $\leq x$ and $\lceil x \rceil$ denotes the least integer $\geq x$.

Properties of LFSR Sequences

1. The r taps $q_1, q_2, \cdots, q_r$ on the cells of an r-stage LFSR correspond to a *connection polynomial*

$$q(X) = q_r X^r + q_{r-1} X^{r-1} + \cdots + q_1 X - 1$$

with coefficients q_i in $\mathbf{Z}/(2)$. The period (and many other properties) of the LFSR sequence may be expressed in terms of the Galois theory of this polynomial.

2. Suppose $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ is a periodic sequence of bits obtained from a linear feedback shift register of length r with connection polynomial $q(X)$. If $q(X)$ is irreducible, and if $\gamma \in GF(2^r)$ is a root of $q(X)$ then for all $i = 0, 1, 2, \cdots$ we have,

$$a_i = Tr(A\gamma^i)$$

for some $A \in GF(2^r)$ (which corresponds to the choice of initial loading of the shift register). Here, $Tr : GF(2^r) \to GF(2)$ denotes the trace function.

3. Any infinite binary sequence $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ may be identified with its generating function, $A(X) = \sum_{i=0}^{\infty} a_i X^i$ which is an element of the ring $\mathbf{Z}/(2)[[X]]$ of formal power series with coefficients in the integers modulo 2. It is well known (cf. [13]) that *the sequence* **a** *is eventually periodic if and only if its generating function is equal to a quotient of two polynomials,*

$$A(X) = r(X)/q(X) \in \mathbf{Z}/(2)[[X]]$$

in which case the denominator $q(X)$ is the connection polynomial for a linear feedback shift register which generates the periodic part of the sequence **a**. The sequence **a** is *strictly* periodic if and only if $\deg(r) < \deg(q)$.

4. The size of the smallest LFSR that generates a given periodic sequence **a** is called the *linear complexity* or *equivalent linear span* of **a**; it is an important measure of the cryptographic security of the sequence. Such a shift register (of minimum size) may be found in an efficient way using the Berlekamp-Massey algorithm, which may be interpreted as the continued fraction expansion of the fraction $r(X)/q(X)$ in the ring $\mathbf{Z}/(2)[[X]]$ of formal power series. This algorithm is optimal in two senses: (a) it determines the *smallest* LFSR whose output coincides with **a**, and (b) it does so with minimal information: only the first $2 \cdot \text{span}(\mathbf{a})$ bits of the sequence **a** are needed.

5. The generating function of the bitwise sum of two binary pseudorandom sequences $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ and $\mathbf{b} = (b_0, b_1, b_2, \cdots)$ is given by addition $C(X) = A(X) + B(X) \in \mathbf{Z}/(2)[[X]]$ in the ring of formal power series. If **a** and **b** are periodic, then so is the bitwise sum **c**, and its equivalent linear span is no greater than the sum of the linear spans of **a** and **b**.

6. An m-sequence is a LFSR sequence of maximum possible period $T = 2^r - 1$ (where the shift register has r stages). It is a remarkable but well known fact that the m-sequences are exactly those sequences generated by LFSRs whose taps correspond to *primitive* connection polynomials. Such sequences are balanced and have the de Bruijn property: in any single period of an $m-$sequence, every nonzero binary string of length r occurs exactly once. The autocorrelation function of an m-sequence is two valued; the out-of-phase values are all equal to -1.

7. The $2^n - 1$ cyclic permutations of a single period of an m-sequence form the nonzero codewords of a ("punctured") first order cyclic Reed-Muller code. These codes are of fundamental importance in coding theory and are prototypes of the general "finite geometry" codes.

Properties of FCSR Sequences

We now describe the main definitions and results of this paper.

1′. The r taps $q_1, q_2, \cdots, q_r$ on the cells of an r-stage FCSR define a *connection integer* (cf. Definition 3.1)

$$q = q_r 2^r + q_{r-1} 2^{r-1} + \cdots + q_1 2 - 1.$$

The period (and many other properties) of the FCSR sequence may be expressed in terms of number-theoretic properties of this integer.

2′. In §6 we prove that if a periodic sequence $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ is generated by a FCSR with connection integer q, and if $\gamma = 2^{-1} \in \mathbf{Z}/(q)$ is the inverse of 2 in the cyclic group of integers modulo q, then there exists $A \in \mathbf{Z}/(q)$ such that for all $i = 0, 1, 2, \cdots$ we have,

$$a_i = A\gamma^i \pmod q) (\bmod 2).$$

3′. Any infinite binary sequence $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ may be identified with the formal power series, $\alpha = \sum_{i=0}^{\infty} a_i 2^i$ which is an element of the ring $\mathbf{Z}_2$ of 2-adic numbers (cf.§2). The sequence $\mathbf{a}$ is eventually periodic if and only if the 2-adic number α is rational, i.e. if there exist integers r, q such that

$$\alpha = r/q \in \mathbf{Z}_2.$$

In this case, the denominator q is the connection integer for a FCSR which generates the periodic part of the sequence $\mathbf{a}$ (cf. Theorem 4.1.) The sequence $\mathbf{a}$ is *strictly* periodic if and only if $\alpha < 0$ and $|r| < |q|$ (cf. Corollary 4.2).

4′. The size of the smallest FCSR that generates a given periodic sequence $\mathbf{a}$ we call the *2-adic span* of $\mathbf{a}$; (cf. Definition 9.1). Such a shift register (of minimum size) may be found in an efficient way using 2-adic approximation theory (cf. Section 10). Our algorithm is optimal in both of the above senses: it determines the *smallest* FCSR whose output coincides with $\mathbf{a}$, and it does so with knowledge of only $2M + 2\log(M)$ bits of the sequence $\mathbf{a}$ (where M denotes the 2-adic span of $\mathbf{a}$). Although our algorithm is based on de Weger's theory ([47]) of approximation lattices, it differs from de Weger's algorithm in that ours is *adaptive*: each time a new bit is determined (say, by a known plaintext attack), it is used to quickly update the previously determined FCSR. Thus, the number of bits need not be known ahead of time.

5′. Suppose two infinite, periodic sequences $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ and $\mathbf{b} = (b_0, b_1, b_2, \cdots)$ are added with the *carry* operation. This process is called the summation combiner; it was invented by Massey and Rueppel [37, 44], and was suggested as a means for generating cryptographically secure binary sequences from insecure ones. The resulting sequence $\mathbf{c} = (c_0, c_1, c_2, \cdots)$ is given by addition $\gamma = \alpha + \beta \in \mathbf{Z}_2$ in the ring of 2-adic integers (where $\gamma = \sum_{i=0}^{\infty} c_i 2^i$) (cf. Section 2). In Theorem 9.5 we prove that the 2-adic span of the sequence $\mathbf{c}$ is approximately bounded by the sum of the 2-adic spans of $\mathbf{a}$ and $\mathbf{b}$.

6′. An ℓ-sequence is an FCSR sequence of maximum possible period $T = q - 1$ (where q is the connection integer of the FCSR). The ℓ-sequences are generated by FCSR's with connection integers q for which 2 is a *primitive root*. A single period of an ℓ sequence is a cyclic shift of the sequence formed by *reversing* a single period of the binary expansion of the fraction $1/q$. These sequences have been studied since the time of Gauss (([12], [3, Theorem 1], or [44, p. 219].) They have remarkable distribution and correlation properties (cf. Section 13) which are parallel to those of m-sequences.

7′. The $q-1$ cyclic permutations of a single period of an ℓ-sequence form the nonzero codewords of a Barrows-Mandelbaum [2, 33] arithmetic code. The generation of these codes using FCSR circuitry is new.

In section 11 we present a method for attacking the summation cipher. Suppose two periodic binary sequences **a** and **b** are summation-combined to give a binary sequence **c**. Although the linear span of **c** approaches the *product* of the linear spans of **a** and **b**, it follows from (**5′**.) above that the 2-adic span of **c** is only of the order of the *sum* of the 2-adic spans of **a** and **b**. Furthermore, the rational approximation algorithm (**4′**) above will find an FCSR which generates the sequence **c** with knowledge (approximately) $2 \cdot \mathrm{span}_2(\mathbf{c})$ bits.

There is a huge variety of possible variations on the "feedback with carry" theme, some of which we present at the end of this paper.

It is remarkable that sequences generated with the FCSR architecture can be analyzed at all (although there still remains a number of interesting questions). It is even more remarkable that this simple feedback circuitry leads immediately to a variety of deeper number-theoretic issues. We believe that FCSR sequences (and their generalizations) are likely to find many applications in stream cipher technology.

Related Literature.

The authors have published announcements (with sketch of proofs, or with no proofs at all) of some of these results, in various conference proceedings: [23], [24], [25], [26]. Our shift registers are nicely described in §17.4 and §17.5 of [45], where various architectures for combining them with other shift register sequences are suggested. See also [10] (to appear). We also wish to draw the reader's attention to the subsequent developments [22] and [27].

The closely related paper of Marsaglia and Zaman [36] (cf [35]) was recently brought to our attention: their random number generator may be described as an FCSR with two taps, whose cells contain integers modulo b (rather than modulo 2). (Here, b is some large integer.) Thus there is some overlap between their analysis and ours. In particular, Marsaglia and Zaman prove (a) that the period of their generator is given by the order of b modulo $q = b^r + b^s - 1$ (where the taps occur on cells number r and s), and (b) that the output sequence of their generator may be identified with the b-adic expansion of a certain rational number a/q. These are the analogues of our Corollary 2.2 and Theorem 4.1 respectively.

We also learned of the (apparently unpublished) manuscript [1] in which the Euclidean algorithm is proposed as a possible method for the efficient prediction of the Marsaglia-Zaman generator. We have relied heavily on the p-adic approximation theory of [47, 32]. Related results appear in [14, 28, 34].

Another important measure of (nonlinear) complexity is the "maximum order complexity" ([19, 20, 21]) and its determination using the Blumer algorithm ([4]). This is discussed briefly in section 9 however very little is known about the relationship between 2-adic complexity and maximum order complexity.

The summation combiner was previously shown to be vulnerable to the "correlation attack" of Meier and Staffelbach [38, 39].

Acknowledgements.

We wish to thank Hugh Williams for his help in tracking down the various conjectures and results on primes q with large $\mathrm{ord}_q(2)$ which are discussed in Section 13. We are grateful to Bruce Schneier for directing our attention to a number of important implementation issues, especially those concerning degenerate initial loadings. We have profited from useful conversations with Mark McConnell, and we would like to thank two anonymous referees for their careful reading and many thoughtful comments on an earlier version of this paper. The second author would like to thank the Institute for Advanced Study in Princeton N. J. for their hospitality and support (partially provided by the Ellentuck Fund and National Science Foundation grant # DMS 9304580) while this paper was being revised.

2 Review of 2-adic Numbers

In this section we briefly review some basic facts about 2-adic numbers, and fix a notation for the 2-adic numbers; the interested reader may wish to consult one of the many excellent references on p-adic numbers, for example, [11], [30], or [29] Section 4.1, ex. 31.

A 2-adic integer is a formal power series $\alpha = \sum_{i=0}^{\infty} a_i 2^i$, with $a_i \in \{0, 1\}$. Such a power series does not converge in the usual sense, but it can nevertheless be manipulated as a formal object; the collection of all such power series forms the ring $\mathbf{Z}_2$ of 2-adic integers. The main difference between the ring $\mathbf{Z}_2$ and the ring $\mathbf{Z}/(2)[[X]]$ of formal power series in X, is that addition in $\mathbf{Z}_2$ is performed by "carrying" overflow bits to higher order terms, so that $2^i + 2^i = 2^{i+1}$. Multiplication is defined by shift and add. Using these operations, $\mathbf{Z}_2$ becomes a ring with additive identity 0 and multiplicative identity $1 = 1 \cdot 2^0$. (Some readers may find it less confusing to think of formal power series in some indeterminate, Y, rather than 2, and to use the rule $Y^i + Y^i = Y^{i+1}$. It turns out that the use of the number 2 instead of Y facilitates many computations and comparisons between the 2-adic integers and the usual integers.)

The first surprising fact is that the number -1 is represented by

$$-1 = 1 + 2^1 + 2^2 + 2^3 + \cdots$$

as may be verified by adding 1 to both sides of the equation. It follows that $\mathbf{Z}_2$ contains *all* the integers. The negative integer $-q$ is associated to the product

$$-q = (1 + 2 + 2^2 + 2^3 + \cdots)(q_0 + q_1 2 + \cdots + q_r 2^r) \tag{1}$$

The inclusion $\mathbf{Z} \subset \mathbf{Z}_2$ is compatible with addition and multiplication. There is a formula for multiplication by -1 which is much simpler than equation (1). If $\alpha = 2^r(1 + \sum_{i=0}^{\infty} a_i 2^i)$ then

$$-\alpha = 2^r(1 + \sum_{i=0}^{\infty} \bar{a}_i 2^i) \tag{2}$$

where $\bar{a}_i$ denotes the complementary bit. (Just check that $\alpha + -\alpha = 0$.) In other words, the bit sequence for $-\alpha$ is obtained by keeping any leading string of 0's as well as the first nonzero bit, then by complementing all subsequent bits.

In the integers, only $+1$ and -1 have integer inverses. However, in $\mathbf{Z}_2$, any formal power series

$$\alpha = 1 \cdot 2^0 + a_1 2^1 + a_2 2^2 + \cdots$$

which begins with 1 has a (multiplicative) inverse,

$$\alpha^{-1} = 1 \cdot 2^0 + b_1 2^1 + b_2 2^2 + \cdots$$

as may be verified by long division. In particular, every odd integer $q \in \mathbf{Z}$ has a unique inverse in $\mathbf{Z}_2$. Thus, the ring $\mathbf{Z}_2$ contains every rational number p/q provided q is odd. We make this explicit in the following statement:

Theorem 2.1 *There is a one to one correspondence between rational numbers $\alpha = p/q$ (where q is odd) and eventually periodic binary sequences $\mathbf{a}$, which associates to each such rational number α the bit sequence $(a_0, a_1, a_2, \cdots)$ of its 2-adic expansion. The sequence $\mathbf{a}$ is strictly periodic if and only if $\alpha \leq 0$ and $|\alpha| < 1$.*

Proof: Although the proof is standard, we include it here because it is the basis for many of the results in this paper. Let us first consider the *strictly* periodic case. Let $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ be a strictly periodic sequence of period T. Set $\alpha = \sum_{i=0}^{\infty} a_i 2^i$. Computing in $\mathbf{Z}_2$, we find

$$2^T \alpha = \sum_{i=0}^{\infty} a_i 2^{i+T} = \sum_{i=0}^{\infty} a_{i+T} 2^{i+T} = \sum_{i=T}^{\infty} a_i 2^i = \alpha - \sum_{i=0}^{T-1} a_i 2^i$$

Hence

$$\alpha = -\frac{(\sum_{i=0}^{T-1} a_i 2^i)}{(2^T - 1)} \tag{3}$$

is a negative rational number. Let us write $\alpha = p/q$ as a fraction reduced to lowest terms with q positive. Then q is odd, $p \leq 0$, and $|p| < q$.

On the other hand, suppose that $\alpha = p/q$ is given in lowest terms with q an odd positive integer, $p \le 0$ and $|p| < q$. Let $T = \mathrm{ord}_q(2)$ be the smallest integer such that $2^T \equiv 1 \pmod{q}$. Such a T exists because q is odd. Then $2^T - 1$ is divisible by q, so set $s = (2^T - 1)/q$, and write $s \cdot (-p) = \sum_{i=0}^{T-1} a_i 2^i$. Thus $\alpha = sp/(2^T - 1)$. The calculations leading to equation (3) may be run backwards to see that the segment $a_0, a_1, \cdots, a_{T-1}$ is a single period of a strictly periodic sequence.

Now suppose that $\alpha = p/q$ is an *arbitrary* rational number. Let $M = \lceil \alpha \rceil$ be the next largest integer. If $M \ge 0$ then its 2-adic expansion is finite (i.e. it ends in an infinite string of 0's). If $M < 0$ then its 2-adic expansion ends in an infinite string of 1's (cf. equation 2). But $\alpha = M + p'/q$ where $p' \le 0$ and $|p'| < q$, so the 2-adic expansion for p'/q is periodic. It follows that the 2-adic expansion for the sum $\alpha = M + p'/q$ is eventually periodic (in fact it is easy to see that the 2-adic sum of any two eventually periodic sequences is again eventually periodic).

On the other hand, an eventually periodic sequence $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ corresponds to a rational number $\alpha = \sum_{i=0}^{\infty} a_i 2^i$ because it is given by a finite transient term $\sum_{i=0}^{K-1} a_i 2^i$ (for some nonnegative integer K) plus a periodic term, $\sum_{i=K}^{\infty} a_i 2^i = 2^K \sum_{j=0}^{\infty} a_{j+k} 2^j$, both of which are rational numbers. □

One simple consequence of the proof of Theorem 2.1 is the following old result of Gauss ([12], [3, Theorem 1], or [44, p. 219].)

Corollary 2.2 *If p and q are relatively prime, $-q < p \le 0$, and q is odd, then the period of the bit sequence for the 2-adic expansion of $\alpha = p/q$ is $T = ord_q(2)$.* □

3 Feedback Shift Registers with Carry

In this section and the next, we give the definition and derive the basic properties of FCSR's. Throughout this section, we fix an odd positive integer $q \in \mathbf{Z}$ and let $r = \lfloor \log_2(q+1) \rfloor$ (where $\lfloor\ \ \rfloor$ denotes the floor or integer part). Write

$$q + 1 = q_1 2 + q_2 2^2 + \cdots + q_r 2^r \tag{4}$$

for the binary representation of the integer $q+1$ (so $q_r = 1$). The shift register uses r stages and $\lfloor \log_2(r) \rfloor$ additional bits of memory (or less: see below). The feedback connections are given by the bits $\{q_1, q_2, \cdots, q_r\}$ appearing in equation (4).

Definition 3.1 *The FCSR with connection integer q is the register depicted in Figure 2.*

(Notice that $q_0 = -1$ does not correspond to a feedback tap, and that the coefficients of high powers of 2 are close to the output cell.) In Figure 2, Σ denotes integer addition. The contents of the register at any given time consists of r bits, denoted $a_{n-1}, a_{n-2}, \cdots, a_{n-r+1}, a_{n-r}$. The operation of the shift register is defined as follows:

A1. Form the integer sum $\sigma_n = \sum_{k=1}^{r} q_k a_{n-k} + m_{n-1}$.

A2. Shift the contents one step to the right, outputting the rightmost bit a_{n-r}.

A3. Place $a_n = \sigma_n \pmod 2$ into the leftmost cell of the shift register.

A4. Replace the memory integer m_{n-1} with $m_n = (\sigma_n - a_n)/2 = \lfloor \sigma_n/2 \rfloor$.

We refer to q as the *connection integer* because its binary expansion gives the analog to the connection polynomial in the usual theory of linear feedback shift registers.

Implementation.

Fast hardware implementation of this operation may be simplified by using a ripple adder and by incorporating the memory m into the shift register as indicated in Figure 3. (In this figure, individual memory bits are labeled m^2, m^1, m^0 although the memory could, of course, be much larger. The symbol Σ denotes integer sum and FA. denotes a full adder.) The operation of the shift register may be described by replacing (A1),..., (A4) with the following steps (B1),...,(B3) which are easily seen to be equivalent.

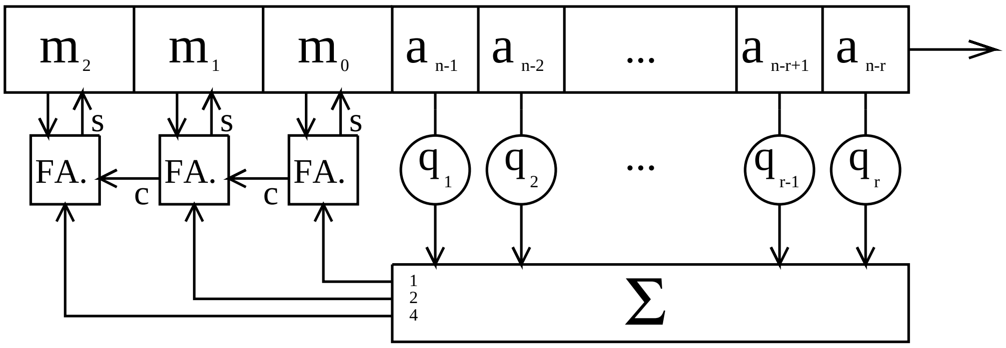

Figure 3: Shift register with ripple adder

B1. Form the integer sum $\sigma'_n = \sum_{k=1}^{r} q_k a_{n-k}$.

B2. Add σ'_n to the memory contents $m_{n-1} = \sum m^i 2^i$ in the ripple adder.

B3. Move the answer back into the memory portion of the shift register.

B4. Shift the whole register one step to the right, outputting the rightmost bit a_{n-r}.

Memory Requirements.

The fast hardware generator in figure 3 is limited to nonnegative values of the memory integer m. Moreover, we will see in corollary 4.2 that *any strictly periodic sequence of bits may be generated by an FCSR using only nonnegative memory values.* However, in order to efficiently generate a given *eventually* periodic sequence, it may be necessary to allow negative values for the memory. Moreover the analysis in sections 4 to 10 is valid whether the memory is positive or negative. So, for the remainder of the paper we will consider FCSR architectures with signed memory values.

Let us consider an r-stage FCSR with odd positive connection integer $q = -1+q_1 2+\ldots+q_r 2^r$. Let $w = wt(q+1)$ be the number of nonzero q_i, $i = 1, \cdots, r$, the Hamming weight of $q+1$. A *state* of the FCSR is a specification of the memory m and of all the cell contents. We shall say that a state is *periodic* if, left to run, the FCSR will eventually return to that same state.

Proposition 3.2 *If an FCSR is in a periodic state then the memory is in the range* $0 \le m < w$ *(which may therefore be accomplished by using no more than* $\lfloor \log_2(w-1) \rfloor + 1$ *bits of memory). If the initial memory* $m_{n-1} \ge w$ *then it will monotonically decrease and will arrive in the range* $0 \le m < w$ *within* $\lfloor \log_2(m_{n-1} - w) \rfloor + r$ *steps. If the initial memory* $m_{n-1} < 0$ *then it will monotonically increase and will arrive in the range* $0 \le m < w$ *within* $\lceil \log_2(|m_{n-1}|) \rceil + r$ *steps.*

Proof: First, observe that if the initial memory value m_{n-1} lies in the range $0 \le m_{n-1} < w$ then the same will be true for all later values of the memory. This follows from (A1) and (A4) because $\sigma_n \le w + m_{n-1} < 2w$ so $m_n = \lfloor \sigma_n/2 \rfloor < w$.

By the same argument, if the initial memory value is $m_{n-1} = w$ then later values of memory will be no greater than w; but in this case, within r steps, the memory will drop below w (and will remain so thereafter) for the following reason. If the memory does not decrease (i.e. if $m_n = w$) then this means that a 1 appeared at *all* the tapped cells, that $\sigma_n = 2w$ and that $a_n = \sigma_n (\bmod 2) = 0$ was fed into the register. The value of σ will fall below $2w$ when this 0 reaches the first tapped cell (if not before), at which time we will have $m = \lfloor \sigma/2 \rfloor < w$.

Moreover, if we initialize a FCSR with a larger memory value, $m_{n-1} > w$, then with each step, the excess $e_{n-1} = m_{n-1} - w$ will become reduced by a factor of 1/2, that is, $e_n \le \left\lfloor \frac{1}{2} e_{n-1} \right\rfloor$. So after $\lfloor \log_2(m_{n-1} - w) \rfloor + 1$ steps, the memory will be no more than w. This follows from (A1) and (A4) which give

$$e_n = m_n - w = \lfloor \sigma_n/2 \rfloor - w \le \lfloor (w + w + e_{n-1})/2 \rfloor - w = \lfloor e_{n-1}/2 \rfloor$$

Now let us consider the case of negative initial memory, $m_{n-1} < 0$. By (A1), it is possible that $\sigma_n \ge 0$, in which case the next memory value will be $m_n \ge 0$ (where it will remain thereafter). So let us suppose that $\sigma_n < 0$. Then by (A4), $|m_n| \le (|\sigma_n| + 1)/2 \le (|m_{n-1}| + 1)/2$. Iterating this formula, we find that after $K = \lceil \log_2(|m_{n-1}|) \rceil$ steps, either the memory m has become nonnegative, or else

$$|m| \le \frac{m_{n-1}}{2^K} + \frac{1}{2^K} + \frac{1}{2^{K-1}} + \ldots + \frac{1}{2} < 2$$

in which case the memory must be $m = -1$. There is a single situation in which the memory can remain at -1 forever: if there are no feedback taps on the shift register (so $q = -1$). In this case the memory will feed 1's into the shift register forever. However we assumed that $q > 0$ to rule out this possibility. If $q > 0$, then as soon as a nonzero feedback occurs, the memory will become nonnegative, where it will remain thereafter. $\square$

4 Analysis of FCSRs

In this section we use arithmetic in the 2-adic integers in order to determine the output sequence of a given FCSR. Suppose we fix an r-stage FCSR with connection integer $q = -1 + q_1 2 + q_2 2^2 + \cdots + q_r 2^r$, with initial memory m_{r-1}, and with initial loading $a_{r-1}, a_{r-2}, \cdots, a_1, a_0$. (See Figure 2.) The register will generate an infinite, eventually periodic sequence $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ of bits, to which we associate the 2-adic integer $\alpha = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \cdots \in \mathbf{Z}_2$ which we call the *2-adic value* of the FCSR (with its initial loading and initial memory). Define

$$p = \sum_{i=0}^{r-1} \sum_{j=0}^{i} q_j a_{i-j} 2^i - m_{r-1} 2^r, \tag{5}$$

where we have set $q_0 = -1$ so that $q = \sum_{i=0}^{r} q_i 2^i$.

Theorem 4.1 *The output,* $\mathbf{a}$, *of a FCSR with connection integer $q > 0$, initial memory value m_{r-1}, and initial loading $a_{r-1}, a_{r-2}, \cdots, a_1, a_0$, is the bit sequence of the 2-adic representation of the rational number $\alpha = p/q$, in other words,*

$$\alpha = \sum_{i=0}^{\infty} a_i 2^i = \frac{p}{q} \in \mathbf{Z_2} \tag{6}$$

Theorems 4.1 and 2.1 and proposition 3.2 give the following:

Corollary 4.2 *If* $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ *is an eventually periodic binary sequence then the associated 2-adic number $\alpha = \sum a_i 2^i$ is a quotient of two integers, $\alpha = p/q$ and the denominator q is the connection integer of a FCSR which generates the sequence* $\mathbf{a}$. *The sequence* $\mathbf{a}$ *is strictly periodic if and only if $p < 0$ and $|p| < q$. In this case the values of the memory may be taken to lie in the range $0 \le m < wt(q+1)$.*

Proof of Theorem 4.1. The computations in this section parallel those in [13], but here they take place in $\mathbf{Z}_2$ rather than in $\mathbf{Z}/(2)[[X]]$. Let us consider the transition from one state of the shift register to the next. Suppose that, for some given state, the value of the memory is m_{n-1} and that the contents of the register is given by the r bits $a_{n-1}, a_{n-2}, \cdots, a_{n-r}$, with a_{n-1} the leftmost bit and a_{n-r} the rightmost bit, and where the register shifts towards the right. The next state is determined by calculating (A1)

$$\sigma_n = m_{n-1} + \sum_{i=1}^{r} q_i a_{n-i}, \tag{7}$$

writing the new memory contents as $m_n = \lfloor \sigma_n/2 \rfloor$, and writing the new contents of the leftmost cell as $a_n = \sigma_n \pmod 2$ (see (A3) and (A4)). (The remaining bits are shifted once to the right.) These equations may be combined into the expression

$$\sigma_n = 2m_n + a_n.$$

It follows that

$$a_n = \sum_{i=1}^{r} q_i a_{n-i} + (m_{n-1} - 2m_n), \tag{8}$$

provided that $n \geq r$. Suppose the initial loading of the register consists of memory m_{r-1} and with register bit values $a_{r-1}, a_{r-2}, \cdots, a_1, a_0$. Now substitute (8) into the expression (6) for α to obtain

$$\begin{aligned}\alpha &= a_0 + a_1 2 + \cdots + a_{r-1}2^{r-1} + \sum_{n=r}^{\infty} a_n 2^n \\ &= x + \sum_{n=r}^{\infty}(\sum_{i=1}^{r} q_i a_{n-i})2^n + \sum_{n=r}^{\infty}(m_{n-1} - 2m_n)2^n, \end{aligned} \tag{9}$$

where

$$x = a_0 + a_1 2 + \cdots + a_{r-1}2^{r-1}$$

is the integer represented by the initial loading of the register. The second summation in equation (9) cancels except for the first term, m_{r-1}, leaving

$$\begin{aligned}\alpha &= x + m_{r-1}2^r + \sum_{n=r}^{\infty}\sum_{i=1}^{r} q_i 2^i a_{n-i} 2^{n-i} \\ &= x + m_{r-1}2^r + \sum_{i=1}^{r} q_i 2^i (\sum_{n=r}^{\infty} a_{n-i}2^{n-i}) \\ &= x + m_{r-1}2^r + \sum_{i=1}^{r} q_i 2^i (\alpha - (a_0 2^0 + a_1 2^1 + \cdots + a_{r-i-1}2^{r-i-1})) \\ &= x + m_{r-1}2^r + \alpha \sum_{i=1}^{r} q_i 2^i - \sum_{i=1}^{r-1}\sum_{j=0}^{r-i-1} q_i 2^i a_j 2^j \end{aligned}$$

(where the inner sum is empty, hence zero, when $i = r$ in the third line). These equations give

$$\begin{aligned}\alpha &= \frac{x + m_{r-1}2^r - \sum_{i=1}^{r-1}\sum_{j=0}^{r-i-1} q_i 2^i a_j 2^j}{1 - \sum_{i=1}^{r} q_i 2^i} \\ &= \frac{\sum_{i=0}^{r-1}\sum_{j=0}^{r-i-1} q_i a_j 2^{i+j} - m_{r-1}2^r}{q} \end{aligned} \tag{10}$$

since $q_0 = -1$. The double summation is over all pairs of integers $0 \le i, j \le r-1$ with $i+j \le r-1$. Setting $k = i + j$ gives

$$\alpha \;=\; \frac{\sum_{k=0}^{r-1} \sum_{i=0}^{k} q_i a_{k-i} 2^k - m_{r-1} 2^r}{q} = \frac{p}{q}. \tag{11}$$

as claimed. □

Corollary 4.3 *Changing the memory by b changes the value of α by $-b2^r/q$. If $\alpha = p/q < 0$ then the initial memory $m_{r-1} \ge 0$ is nonnegative.*

Remarks. There are three easy initial loadings which guarantee a strictly periodic output:

1. set $m = 1$ and all the $a_j = 0$ (so $p = -2^r$);
2. set $m = 1$, $a_0 = 1$, and all the other $a_j = 0$, (so $p = q - 2^{r+1}$);
3. set $m = 0$, $a_{r-1} = 1$, and all the other $a_j = 0$, (so $p = -2^{r-1}$).

If $-q < p < 0$ and p is relatively prime to q, then by Corollary 2.2, the period of the sequence is $T = \mathrm{ord}_q(2)$. If p and q have a common factor, then the period is a divisor of $\mathrm{ord}_q(2)$. If $p \ge 0$ or if $p \le -q$ then the sequence has a transient prefix before it drops into a periodic state. It appears to be difficult to determine which positive fractions α may be represented by equation (5) with m_{r-1} nonnegative. Most positive fractions require negative initial values for the memory.

5 Initial Loading

In this section we answer the reverse question: Suppose we are given a fraction $\alpha = p/q$ (with q an odd positive integer), how do we determine a FCSR and initial loading whose output sequence coincides with the 2-adic expansion of α? Set $r = \lfloor \log_2(q+1) \rfloor$. Write $q = \sum_{i=0}^{r} q_i 2^i$ with $q_0 = -1$ and $q_i \in \{0, 1\}$ for $i > 0$. Consider an FCSR with r stages and with connection integer q. The initial memory m_{r-1} and initial loading $a_0, a_1, \ldots, a_{r-1}$ are related to p and q by equation (11) which may be solved using the following procedure:

C1. Compute $a_0 + a_1 2 + \cdots + a_{r-1} 2^{r-1} = p/q \bmod 2^r$. (This is the first r bits in the 2-adic expansion for p/q.) In general computing modular quotients is apparently hard. However, when the modulus is a power of a prime they can be computed efficiently. It is straightforward to do so in time $\mathcal{O}(r^2)$.

C2. Compute $y = \sum_{i=0}^{r-1} \sum_{j=0}^{i} q_j a_{i-j} 2^i$, say by a modified multiplication algorithm.

C3. Compute $m = (y - p)/2^r$ in time $\mathcal{O}(r)$.

Use $a_0, \cdots, a_{r-1}$ as the initial loading and m as the initial memory in a FCSR with connection integer q. This FCSR will output the 2-adic expansion of p/q.

Degenerate initial loadings.

Let us say that an initial loading is degenerate if the 2-adic number $\alpha = p/q$ corresponding to the output sequence is an integer (in the usual sense). In this case, after a transient prefix, the FCSR outputs all 0's (if $\alpha > 0$) or all 1's (if $\alpha < 0$). It is easy to see that there are only two possible degenerate final states: (a) $m = 0$ and all $a_i = 0$ and (b) $m = \mathrm{wt}(q+1) - 1$ and all $a_i = 1$. But how long can the prefix be? (We wish to thank the referee for helping us to sharpen the following statement.)

Theorem 5.1 *Consider an r-stage FCSR with a nontrivial tap on the last cell (i.e. with $q_r = 1$). Suppose the initial loading is degenerate. If the initial memory $m > 0$ is positive then the output will stabilize to all 1's after no more than $\lceil \log_2(1+m) \rceil$ steps (and it cannot stabilize to all 0's). If the initial memory $m < 0$ then the output cannot stabilize to all 1's: if the register has at least two taps, then the output will stabilize to all 0's within $\lceil \log_2(wt(q+1) + |m| - 1) \rceil$ steps; if it has only one tap then the output will stabilize to all 0's within $\lceil \log_2(|m|2^r/(2^r - 1)) \rceil$ steps. If the initial memory $m = 0$ then the only degenerate initial loading is the trivial one consisting of all 0's.*

Proof: Suppose the value $\alpha = p/q$ of the FCSR is an integer. First we consider the case that the initial memory is $m \geq 0$. Let us consider the possibilities $\alpha \geq 0$ and $\alpha \leq 0$ separately.

If $\alpha \geq 0$ is an integer, eventually the FCSR will output all 0's. But this is not possible: before the sequence has become all zero, the memory must have been cleared (otherwise, once the register has cleared, it would eventually feed a 1 back into the register). Now if the memory is cleared, when the last 1 in the register passes the last tap, it will feed a 1 back to the beginning of the register. So the register will never be cleared.

If $\alpha < 0$ is an integer, then eventually the FCSR will output all 1's. Since $m \geq 0$ and $p < 0$ we have by equation (10), $|p| \leq x + m2^r \leq (1+m)2^r$. If $q \neq 2^r - 1$ then $q > 2^r$ and we conclude that $|\alpha| < 1 + m$. The output, which is the 2-adic expansion of the integer α becomes all 1's within $\lceil \log_2(1+m) \rceil$ steps by equation (2). A special argument must be made when $q = 2^r - 1$.

Now consider the case of initial memory $m \leq 0$. We claim that the output string cannot degenerate to all 1's: if so, then $\alpha < 0$ so $p < 0$. But by equation (10) the only negative contribution to p is from x, and $x < q$. So if $p < 0$ then $|\alpha| \leq x/q < 1$ which therefore cannot be an integer (other than 0).

Finally, if the initial memory $m < 0$ and if the output stream degenerates to all 0's then $\alpha > 0$ and $p > 0$. It follows from equation (10) (cf. lemma 9.4) that

$$p \leq (|m| + \mathrm{wt}(q+1) - 1)2^r \tag{12}$$

Provided $q \neq 2^r - 1$, we have $q > 2^r$, so $\alpha < \mathrm{wt}(q+1) + |m| - 1$. So in this case, the output sequence is the (reverse of the) binary expansion for α, which takes $\lceil \log_2(\mathrm{wt}(q+1) + |m| - 1) \rceil$ digits, after which we have all 0's. (In the case of a single tap, $q = 2^r - 1$ so $\alpha = |m|2^r/(2^r - 1)$.)
□

6 Exponential Representation of FCSR Sequences

One of the most powerful techniques for the analysis of shift register sequences is its exponential representation. Suppose $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ is a periodic sequence of bits obtained from a linear feedback shift register of length r, with connection polynomial $q(X)$. If $q(X)$ is irreducible and if $\gamma \in GF(2^r)$ is a root of $q(X)$ in the finite field with 2^r elements, then for all $i = 0, 1, 2, \cdots$ we have,

$$a_i = Tr(A\gamma^i)$$

for some $A \in GF(2^r)$ (which corresponds to the choice of initial loading of the shift register). Here, $Tr : GF(2^r) \to GF(2)$ denotes the trace function. In this section we derive a similar representation for periodic sequences of bits obtained from feedback shift registers with memory.

Theorem 6.1 *Suppose a periodic sequence* $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ *is generated by a FCSR with connection integer* q. *Let* $\gamma = 2^{-1} \in \mathbf{Z}/(q)$ *be the (multiplicative) inverse of* 2 *in the ring* $\mathbf{Z}/(q)$ *of integers modulo* q. *Then there exists* $A \in \mathbf{Z}/(q)$ *such that for all* $i = 0, 1, 2, \cdots$ *we have,*

$$a_i = A\gamma^i \pmod{q} \pmod{2}.$$

Here the notation $(\bmod\ q)(\bmod\ 2)$ means that first the number $A\gamma^i$ should be reduced modulo q to give a number between 0 and $q-1$, and then that number should be reduced modulo 2 to give an element of $\mathbf{Z}/(2)$. (Notice that there is no group homomorphism $\mathbf{Z}/(q) \to \mathbf{Z}/(2)$ if q is odd, so the notation $(\bmod\ q)(\bmod\ 2)$ needs a precise definition.)

Proof Suppose the FCSR is in a state S, meaning the memory has some value m and the register is loaded with bits $a_0, a_1, \cdots, a_{r-1}$. Let us also suppose that the FCSR is in periodic mode, i.e. that the output sequence $\mathbf{a} = (a_0, a_1, \cdots, a_{r-1}, a_r, \cdots)$ is periodic with no transient prefix. Let $T = \mathrm{ord}_q(2)$ denote the period of this sequence (which may be much less than $q-1$). To such a state S we associate its 2-adic value, $f(S)$. By Theorem 4.1, $f(S)$ is a 2-adic integer of the form

$$f(S) = -\frac{p}{q} = \sum_{i=0}^{\infty} a_i 2^i,$$

with $0 \le p \le q-1$. Now let S' denote the next state of the FCSR, so

$$f(S') = -\frac{p'}{q} = \sum_{i=0}^{\infty} a_{i+1} 2^i.$$

Thus, $0 \le p' \le q-1$ and

$$-2\frac{p'}{q} + a_0 = -\frac{p}{q},$$

or $p = 2p' - a_0 q \in \mathbf{Z}$. If we read this equation modulo 2, we see

$$p \equiv a_0 \pmod{2}.$$

Reading this equation modulo q we obtain

$$p' \equiv 2^{-1}p \ (\bmod q).$$

This shows that the sequence of numerators $(p, p', \cdots)$ is obtained by multiplying by γ and reducing mod q, and that the sequence of bits $(a_0, a_1, \cdots)$ is obtained by reducing the numerators modulo 2. Finally, the initial state is arbitrary and given by the choice of some $A \in \mathbf{Z}/(q)$. $\square$

Although Peterson and Weldon [41] consider only the case where q is prime and 2 is a primitive element modulo q, their proof of Theorem 15.5 (p. 458) may be used in this situation to give another proof of Theorem 6.1. The proof presented here is useful because it extends verbatum to various generalizations of the FCSR architecture which involve ramified field extensions of the 2-adic numbers.

7 Example

Let us consider the FCSR with connection integer $q = 37 = 32 + 4 + 2 - 1$. Then we have a 5 stage shift register with feedback connections on the first, second, and fifth cells, counting from the left. The element $\gamma = 2^{-1} \in \mathbf{Z}/(37)$ is $\gamma = 19$.

mem	register	a_0	p	n
0	10011	1	1	0
1	01001	1	19	1
1	10100	0	28	2
1	01010	0	14	3
1	00101	1	7	4
1	00010	0	22	5
0	10001	1	11	6
1	01000	0	24	7
1	00100	0	12	8
0	10010	0	6	9
0	11001	1	3	10
1	11100	0	20	11
1	11110	0	10	12
1	11111	1	5	13
2	01111	1	21	14
2	00111	1	29	15
1	10011	1	33	16
1	11001	1	35	17

mem	register	a_0	p	n
2	01100	0	36	18
1	10110	0	18	19
1	01011	1	9	20
1	10101	1	23	21
1	11010	0	30	22
1	11101	1	15	23
2	01110	0	26	24
1	10111	1	13	25
1	11011	1	25	26
2	01101	1	31	27
2	00110	0	34	28
1	00011	1	17	29
1	00001	1	27	30
1	00000	0	32	31
0	10000	0	16	32
0	11000	0	8	33
1	01100	0	4	34
1	00110	0	2	35

Table 1: The states of a FCSR with $q = 37$.

In Table 1 we consider the initial loading such that the output sequence is given by

$$a_n = \gamma^n \ (\bmod 37)(\bmod 2)$$

for $n = 0, 1, 2, \cdots$, i.e. with the constant $A = 1$, in the notation of the preceding section. The index n is recorded as the last column of Table 1. The column "mem" indicates the integer value of the memory, and a_0 represents the output bit (i.e. the rightmost bit in the register). Each state S of the shift register corresponds to a rational number $f(S) = -p/37$ and the numerator p is recorded also in the table. The table therefore lists all the strictly periodic states of the FCSR.

8 Division by q in the 2-adic integers

Consider the effect of driving a FCSR by introducing an input sequence $\mathbf{b} = (b_r, b_{r+1}, b_{r+2}, \cdots)$ to another input terminal of the adder Σ in Figure 2. The generating function analysis in Section 4 is easily modified to incorporate this driving signal. Equation (7) becomes

$$\sigma_n = b_n + m_{n-1} + \sum_{i=1}^{r} q_i a_{n-i}$$

and equation (10) becomes

$$\alpha = \frac{\beta + x + m_{r-1}2^r - \sum_{i=1}^{r}\sum_{j=0}^{r-i-1} q_i 2^i a_j 2^j}{1 - \sum_{i=1}^{r} q_i 2^i},$$

where $\beta = \sum_{i=r}^{\infty} b_i 2^i \in \mathbf{Z}_2$ is the 2-adic number represented by the driving signal $\mathbf{b}$. This proves,

Theorem 8.1 *A FCSR with connection integer q and initial loading $a_i = 0$ $(i = 0, 1, \cdots, r-1)$ and initial memory $m_{r-1} = 0$ which is driven by a signal $\mathbf{b} = b_0, b_1, b_2, \cdots$ has an output sequence which is given by the 2-adic integer $\alpha = \beta/q$.*

Thus, if we interpret the Z-transform as a 2-adic number (rather than as a formal power series), then the transfer "function" becomes interpreted as division by q. The analogous result in the linear theory is the following: a LFSR with connection polynomial $q(X)$ and with initial loading 0 which is driven by a signal $\beta(X) = b_r X^r + b_{r+1}X^{r+1} + \cdots$ has an output sequence which is given by the coefficients of the formal power series expansion of $\beta(X)/q(X)$.

9 2-adic Span and Complexity

As in the case of linear span, the 2-adic span of a sequence is intended to measure how large a FCSR is required to output the sequence. In the case of LFSRs, this is given by the number of bits in a register that outputs the sequence, and coincides with the degree of the connection polynomial, i.e., the denominator of the rational function giving the power series whose coefficients are the bits of the sequence.

In the 2-adic case, things are more complex. The number of bits in the connection number coincides with the size of the basic register, but additional space is required for the memory. For purely periodic sequences, this extra memory is small (at most log of the number of bits in the basic register), and if such sequences were our only concern we could ignore it. But an *eventually* periodic sequence may require a considerable amount of memory. We would like to define the 2-adic span of an eventually periodic sequence $\mathbf{a}$ to be the number of bits in the register + memory of an FCSR which outputs the sequence a, however even this definition must be approached with care because the memory value may grow as the FCSR runs (see the discussion at the end of this section). In the following paragraph we propose two natural notions: the *span* (an integer which counts the number of bits in the register + memory) and the *complexity* (a real number) of a sequence $\mathbf{a}$. If $\mathbf{a}$ is strictly periodic then the number of cells in the basic register (not counting the memory) is $\lceil \text{complexity}(\mathbf{a}) \rceil$. We show that these two complexity measures differ at most by $\log_2(\text{complexity}(\mathbf{a}))$. From the mathematical point of view, the complexity is the more natural number; from the engineering point of view, the span is the more natural number.

Let $\mathbf{a} = (\mathbf{a_0}, \mathbf{a_1}, \ldots)$ be an eventually period sequence of bits. Suppose an FCSR with connection integer $q = -1 + q_1 2^1 + \ldots + q_r 2^r$ and initial memory m ouputs this sequence, and that $q_r = 1$ (i.e. that $r = \lfloor \log_2(q+1) \rfloor$). We associate to this register the number

$$\lambda = r + \max(\lfloor \log_2(\text{wt}(q+1)) \rfloor + 1, \lfloor \log_2(|m|) \rfloor + 1) + 1$$

of bits in the FCSR, where $\text{wt}(q+1)$ denotes the number of nonzero q_i (for $1 \le i \le r$). (See "memory requirements" in section 3: memory values within the range $0 \le m < \text{wt}(q+1)$ may grow and shrink within this range; memory values outside this range will move montonically toward this range. The second "+1" is a "sign bit" which allows for possible negative memory values.)

Definition 9.1 *The 2-adic span $\lambda_2(\mathbf{a})$ of a binary, eventually period sequence $\mathbf{a} = (a_0, a_1, \cdots)$, is the smallest value of λ which occurs among all FCSR's whose output is the sequence $\mathbf{a}$.*

Now let $\alpha = \sum_{i=0}^{\infty} a_i 2^i = p/q \in \mathbf{Z}_2$ be the fraction in lowest terms whose 2-adic expansion agrees with the sequence $\mathbf{a}$.

Definition 9.2 *The 2-adic complexity of the sequence $\mathbf{a}$ is the real number $\phi_2(\mathbf{a}) = \log_2(\Phi(p,q))$ where $\Phi(p,q) = \max(|p|, |q|)$.*

Proposition 9.3 *If $\alpha = \sum_{i=0}^{\infty} a_i 2^i = p/q$ is the rational number, reduced to lowest terms, corresponding to an eventually periodic sequence $\mathbf{a}$, then the 2-adic span and complexity are related by*

$$|(\lambda_2(\mathbf{a}) - 2) - \phi_2(\mathbf{a})| \le \log_2(\phi_2(\mathbf{a})) \tag{13}$$

Remark. It is also possible to estimate the above quantity in terms of the 2-adic span as follows:

$$|(\lambda_2(\mathbf{a}) - 2) - \phi_2(\mathbf{a})| \le \log_2(\lambda_2(\mathbf{a}) - 2)) + 1. \tag{14}$$

If $\phi_2(\mathbf{a}) \geq 4$ then $\log_2(\phi_2(\mathbf{a})) \leq \phi_2(\mathbf{a})/2$ so (13) gives $\phi_2(\mathbf{a}) \leq 2(\lambda_2(\mathbf{a}) - 2)$, hence $\log_2(\phi_2(\mathbf{a})) \leq 1 + \log_2(\lambda_2(\mathbf{a}) - 2)$, which gives the above inequality. If $\phi_2(\mathbf{a}) < 4$ then the inequality (14) may be checked directly (there are 240 cases with $|p| \leq 15$ and $q \leq 15$).

For notational simplicity, let us write $\lambda = \lambda_2(\mathbf{a})$, $\phi = \phi_2(\mathbf{a})$, $w = \mathrm{wt}(q+1)$ and $\Phi = \Phi(p,q)$. We will need to use the following estimates.

Lemma 9.4 *Suppose $q = -1 + q_1 2 + \ldots + q_r 2^r$ with $q_r = 1$. Let $m \in \mathbf{Z}$ be an integer. Let $a_i \in \{0,1\}$ for $0 \leq i \leq r-1$ and set $x = \sum_{i=0}^{r-1} a_i 2^i$. As in equation (5), (10) define*

$$p = \sum_{i=1}^{r-1} \sum_{j=0}^{r-i-1} q_i a_j 2^{i+j} - x - m2^r \tag{15}$$

Then the double sum is bounded: $\sum_{i=1}^{r-1} \sum_{j=0}^{r-i-1} q_i a_j 2^{i+j} \leq (wt(q+1) - 1)2^r$. Furthermore,

1. *if $p > 0$ and $m < 0$ then*

$$(|m| - 1)2^r < p \leq (|m| + w - 1)2^r \tag{16}$$

2. *if $p < 0$ then $m \geq 0$ and*

$$\max(0, (m - w + 1)2^r) \leq |p| < (m+1)2^r \tag{17}$$

3. *if $p > 0$ and $m \geq 0$ then $m \leq w - 2$ and*

$$p \leq (w - 1 - m)2^r \tag{18}$$

4. *if $p < 0$ then $m < 0$ cannot happen.*

Proof: The proof is a direct calculation,

$$\sum_{i=1}^{r-1} q_i 2^i \sum_{j=0}^{r-i-1} a_j 2^j \leq \sum_{i=1}^{r-1} q_i 2^i (2^{r-i} - 1) \leq \sum_{i=1}^{r-1} q_i 2^r = (\mathrm{wt}(q+1) - 1)2^r$$

since $q_r = 1$ does not appear in the sum. The other estimates follow from this and the fact that $0 \leq x < 2^r$. $\square$

Proof of proposition 9.3 Expanding the absolute value and exponentiating, equation (13) is equivalent to

$$\frac{\Phi}{\log_2(\Phi)} \leq 2^r \max(2^{\lfloor \log(w) \rfloor}, 2^{\lfloor \log(|m|) \rfloor}) \leq \Phi \log_2(\Phi) \tag{19}$$

We do not know a uniform proof for this statement, and there are many cases to consider.

Case 1a: $|p| > q$ and $p < 0$. By equation (15) and (17), we have $m \geq 1$ and $|p| \leq (m+1)2^r$. Note that for $m = 1, 2$ we have $2^{\lfloor \log(m) \rfloor} = m$. So, for $r \geq 2$ we find

$$\frac{|p|}{\log_2(|p|)} \leq \frac{(m+1)2^r}{r} \leq m2^r = 2^r 2^{\lfloor \log(m) \rfloor}$$

which verifies the first inequality in the cases $m = 1, 2$. If $m \geq 3$ and $r \geq 3$ then

$$\frac{|p|}{\log_2(|p|)} \leq \frac{(m+1)2^r}{r} \leq \frac{m2^r}{2} \leq 2^{\lfloor \log(m) \rfloor} 2^r$$

which verifies the first inequality when $r \geq 3$. Finally, if $r = 2$ and $m \geq 3$ then by (17), $\log_2(|p|) \geq r + \log_2(m - w + 1) \geq r + 1$ so

$$\frac{|p|}{\log_2(|p|)} \leq \frac{(m+1)2^r}{r+1} \leq \frac{m2^r}{2} \leq 2^{\lfloor \log(m) \rfloor} 2^r$$

Now consider the second inequality under the same conditions: $p < 0$ and $|p| > q > 2^r$. If $m \leq r$ then

$$2^r 2^{\max(\lfloor \log(m) \rfloor, \lfloor \log(w) \rfloor)} \leq 2^r \max(m, w) \leq 2^r r \leq |p| \log_2(|p|)$$

as desired. So suppose $m \geq r + 1$. Since $p < 0$ we also have $|p| \geq (m - w + 1)2^r$ (cf eq. (17). So

$$\begin{aligned} |p| \log_2(|p|) &\geq (m - w + 1)2^r (r + \log_2(m - w + 1)) \\ &\geq (m - w + 1)2^r (r + 1) \\ &\geq m2^r \geq 2^{\lfloor \log m \rfloor} 2^r \end{aligned}$$

This concludes the proof of proposition 9.3 in the Case 1a.

Case 1b: $|p| > q$ and $p > 0$ If $m \geq 0$ then by (3) we have $m \leq w - 1$. By (18),

$$\frac{p}{\log(p)} \leq \frac{w - 1 - m}{r} 2^r < 2^r \leq 2^{\lambda - 2} \leq 2^r w \leq 2^r r \leq p \log_2(p)$$

If $m < 0$ then

$$\frac{p}{\log_2(p)} \leq \frac{q}{\log_2(q)} \leq 2^r \leq 2^{\lambda - 2} \leq 2^r r \leq p \log_2(p)$$

This concludes the proof in Case 1b.

Case 2a: $|p| \leq q$ and $p < 0$. Since $q \leq 2^{r+1}$,

$$\frac{\Phi}{\log_2(\Phi)} = \frac{q}{\log_2(q)} \leq \frac{2^{r+1}}{r} \leq 2^r \leq 2^{\lambda - 2}$$

which proves the first inequality. By corollary 4.2 the sequence $\mathbf{a}$ is strictly periodic, and the memory m may be taken to lie in the range $0 \leq m \leq w - 1$. In particular, $\max(|m|, w) = w \leq r$. Then

$$2^{\lambda - 2} \leq 2^{r + \lfloor \log w \rfloor} = 2^r w \leq q \log_2 q$$

(This last inequality holds even if $q + 1 = 2^r$ because $w = 1$ in this case.) This concludes the proof in case 2a.

Case 2b: $|p| \leq q$ **and** $p > 0$. The first inequality is proven just as in case 2a above. The second inequality breaks into two further cases: $m \geq 0$ and $m < 0$. First, if $m \geq 0$ then by lemma 9.4 part (3), $m \leq w - 2$ so

$$2^{\lambda-2} = 2^{r+\lfloor \log(w) \rfloor} \leq 2^r w \leq q \log_2(q)$$

(The last inequality even holds if $q + 1 = 2^r$ because $w = 1$.) Next suppose $m < 0$. By (16), $2^{r+1} \geq q \geq p > (|m| - 1)2^r$. Thus, $|m| < 3$, i.e. $m = -1, -2$. Assuming $q \geq 4$ we have

$$2^{\lambda-2} = 2^r \max(w, |m|) \leq q \log_2(q)$$

The cases $q \leq 3$ may be checked separately. □

Proposition 9.3 allows us to relate the 2-adic spans of two sequences to the 2-adic span of their with-carry sum.

Theorem 9.5 *Suppose* **a** *and* **b** *are periodic binary sequences. Let* **c** *denote the binary sequence obtained by adding the sequences* **a** *and* **b** *with carry* (see [37, 44]). *Then the 2-adic complexity of* **c** *is bounded as follows,*

$$\phi_2(\mathbf{c}) \leq \phi_2(\mathbf{a}) + \phi_2(\mathbf{b}) + 1$$

The 2-adic span is bounded as follows,

$$\lambda_2(\mathbf{c}) \leq \lambda_2(\mathbf{a}) + \lambda_2(\mathbf{b}) + 2\log_2(\lambda_2(\mathbf{a})) + 2\log_2(\lambda_2(\mathbf{b})) + 2$$

Proof: Suppose the binary sequences **a** and **b** correspond to 2-adic integers p_1/q_1 and p_2/q_2 respectively. The sum-with-carry sequence **c** corresponds to the 2-adic integer

$$\frac{p_1}{q_1} + \frac{p_2}{q_2} = \frac{p_1 q_2 + p_2 q_1}{q_1 q_2}. \tag{20}$$

so

$$\begin{aligned} \phi_2(\mathbf{c}) &= \log_2(\Phi(p_1 q_2 + p_2 q_1, q_1 q_2)) \\ &\leq \log_2(2\Phi(p_1, q_1)\Phi(p_2, q_2)) = 1 + \phi_2(\mathbf{a}) + \phi_2(\mathbf{b}) \end{aligned}$$

On the other hand, by proposition 9.3

$$\begin{aligned} \lambda_2(\mathbf{c}) - 2 &\leq \phi_2(\mathbf{c}) + \log_2(\phi_2(\mathbf{c})) \\ &\leq \phi_2(\mathbf{a}) + \phi_2(\mathbf{b}) + 1 + \log_2(\phi_2(\mathbf{a}) + \phi_2(\mathbf{b}) + 1) \\ &\leq \lambda_2(\mathbf{a}) + 2\log_2(\lambda_2(\mathbf{a}) - 2) + \lambda_2(\mathbf{b}) + 2\log_2(\lambda_2(\mathbf{b}) - 2) \end{aligned}$$

(by equation (14)), from which the result follows. □

The span may be much less than this if the fraction (20) is not in lowest terms.

What is the 2-adic span of an m-sequence? Although we do not know, it is easy to prove that there exist m-sequences of maximal 2-adic span.

Theorem 9.6 *Suppose* $\mathbf{a}$ *is a periodic sequence with period* $T = 2^N - 1$. *Suppose that* $2^T - 1$ *is prime. Then the 2-adic span of* $\mathbf{a}$ *is* $T + 2$.

Proof: Consider a FCSR which generates the sequence $\mathbf{a}$, and let q denote the connection integer. Then $\mathrm{ord}_q(2) = T = 2^N - 1$. Therefore $2^T \equiv 1 \pmod{q}$. This says that $2^T - 1$ is divisible by q. However, by assumption, $2^T - 1$ is prime, hence $q = 2^T - 1$. The 2-adic span is then at least $\log_2(q+1) + 1 = T + 1$. But any sequence of period T can be generated by a FCSR with T bits in the basic register and one bit of carry (which is always zero) and one sign bit (which is always zero). □

More generally, the same proof shows that the 2-adic span of any periodic sequence with period T is greater than or equal to $\log_2(r+1) + 1$, where r is the smallest prime divisor of $2^T - 1$.

We remark that if $T = 2^N - 1$ and if $2^T - 1$ is prime, then both T and N are prime as well. But the hypotheses of this theorem may be difficult to verify in practice. It is possible that there are only finitely many primes of the form $2^T - 1$, and in any case the largest prime known to date is $q = 2^T - 1$ where $T = 859,433$. An m-sequence generated by a LFSR with only 20 cells already has period $T = 1,048,575$. So, a verification of the hypothesis of the theorem for any larger m-sequence would mean discovering a new prime number.

Complexity Profile. Following Rueppel [44], we would like to define the *2-adic complexity profile* of a pseudorandom sequence $\mathbf{a}$ to be the function $\psi_{\mathbf{a}}$ whose value $\psi_{\mathbf{a}}(k)$ is the 2-adic span of the finite sequence $a_0, a_1, \ldots, a_{k-1}$ consisting of the first k terms of $\mathbf{a}$. However, it is not completely clear how best to define the 2-adic span of a *finite* sequence, since the memory may grow as the FCSR runs. A more meaningful notion may be the *2-adic complexity* of a finite sequence.

Let $\mathbf{a} = a_0, a_1, a_2, a_{k-1}$ be a finite binary sequence. Define

$$\psi(\mathbf{a}) = \log_2(\min_{(p,q)} \Phi(p,q)) = \log_2(\min_{(p,q)} \max(|p|, |q|))$$

where the minimum is taken over all pairs $(p,q) \in \mathbf{Z} \times \mathbf{Z}$ of integers, with q odd, such that the first k bits in the 2-adic expansion of the fraction p/q is precisely $a_0, a_1, \ldots, a_{k-1}$. (In the language of section 10, $\psi_{\mathbf{a}}(k)$ is the minimum value of $\log_2(\Phi(f))$ as f is allowed to vary in the k-th approximation lattice L_k of $\alpha = \sum_{i=0}^{k-1} a_i 2^i$.)

Now let $\mathbf{a} = a_0, a_1, \ldots$ be a possibly infinite binary sequence. The *2-adic complexity profile* $\psi_{\mathbf{a}}(k)$ is the function

$$\psi_{\mathbf{a}}(k) = \psi(a_0, a_1, \ldots, a_{k-1})$$

whose values are the 2-adic complexity of the first k terms in the sequence $\mathbf{a}$. The algorithm presented in section 10 may be used to compute the complexity profile. In fact, (using the notation of figure 4), at the k-th step in the algorithm we have $\psi_{\mathbf{a}}(k) = \log_2(\max(|f|, |g|))$. A highly random sequence $\mathbf{a}$ will exhibit a 2-adic complexity profile $\psi_{\mathbf{a}}(k)$ which grows approximately as $k/2$.

Maximum order complexity. The maximum order complexity of a sequence is the size of the smallest (possibly nonlinear) feedback shift register (without memory) which may be used to generate the sequence (cf. [19, 20, 21, 4]). If a sequence $\mathbf{a}$ is generated by a FCSR with nonnegative memory, then its 2-adic span is no greater than its maximum order complexity. In fact the FCSR of diagram 3 may be interpreted as a (nonlinear) feedback shift register without memory, and the 2-adic span counts the *total* number of cells.

10 Rational Approximation Algorithm

Suppose $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ is an eventually periodic binary sequence. We wish to consider the problem of finding a FCSR whose output sequence coincides with $\mathbf{a}$. First recall that the analogous problem for LFSRs is completely solved by the Berlekamp-Massey algorithm. This algorithm is optimal in two senses:

1. It determines the *smallest* LFSR whose output coincides with $\mathbf{a}$.
2. It does so with minimal information: only the first $2 \cdot \mathrm{span}(\mathbf{a})$ bits of the sequence $\mathbf{a}$ are needed.

Furthermore, the algorithm is adaptive. Each time a new bit is determined (say by a known plaintext attack), it can be used to quickly update the previously determined LFSR. Thus the number of available bits does not need to be known ahead of time.

The Berlekamp-Massey algorithm is equivalent to finding the continued fraction expansion in the field $\mathbf{Z}/(2)[[X]]$ of formal power series, of the element $A(X) = \sum_{i=0}^{\infty} a_i X^i \in \mathbf{Z}/(2)[[X]]$ [7, 9, 40, 48]. One might hope that the continued fraction expansion in the field $\mathbf{Q}_2$ of 2-adic numbers of the element $\alpha = \sum_{i=0}^{\infty} a_i 2^i$ would exhibit similar optimality properties, but this is false. In fact, the continued fraction expansion may fail to converge properly (cf. [47]). There are a number of decoding algorithms available in the context of Hensel and arithmetic codes [14, 28, 34]. However there seem to be no published proofs that any of these algorithms satisfy both of the optimality properties mentioned above.

In this section we present an analog of the Berlekamp-Massey algorithm which has both optimality properties: It constructs the smallest FCSR which generates the sequence $\mathbf{a}$ (theorem 10.1), and it does so using only a knowledge of the first $2M + 2\log M$ bits, where M is the 2-adic span of $\mathbf{a}$ (theorem 10.2). Our algorithm is based on p-adic approximation theory. It is a modification of the procedure outlined by de Weger [47] and Mahler [32], which has the advantage that it is adaptive in the same sense as the Berlekamp-Massey algorithm. (De Weger's algorithm is recalled in section 14 where it is applied to pseudorandom sequences with base p.)

For any pair of integers $f = (f_1, f_2) \in \mathbf{Z} \times \mathbf{Z}$, we write

$$\Phi(f) = \max(|f_1|, |f_2|).$$

Assume we have consecutive terms $a_0, a_1, \cdots$ of a binary sequence $\mathbf{a}$ which is the 2-adic expansion of a number α. We wish to determine a pair of integers $f = (f_1, f_2)$ so that $\alpha = f_1/f_2$

and so that $\Phi(f)$ is minimal among all such pairs of integers. The corresponding FCSR may then be constructed as described in section 5. In the rational approximation algorithm, given in Figure 10, and in the rest of this section, if $f = (f_1, f_2)$ is a pair of integers and if $d \in \mathbf{Z}$ is an integer, write $df = (df_1, df_2)$.

Rational_Approximation
  begin
  input a_i's until the first nozero a_{k-1} is found
  $\alpha = a_{k-1} \cdot 2^{k-1}$
  $f = (0, 2)$
  $g = (2^{k-1}, 1)$
  while there are more bits **do**
    input a new bit a_k
    $\alpha = \alpha + a_k 2^k$
    if $\alpha \cdot g_2 - g_1 \equiv 0 \pmod{2^{k+1}}$ **then**
      $f = 2f$
    else if $\Phi(g) < \Phi(f)$ **then**
      Let d be odd and minimize $\Phi(f + dg)$
      $\langle g, f\rangle = \langle f + dg, 2g\rangle$
    else
      Let d be odd and minimize $\Phi(g + df)$
      $\langle g, f\rangle = \langle g + df, 2f\rangle$
    fi fi
    $k = k + 1$
  od
  return g
  end

Figure 4: Rational Approximation Algorithm for 2-Adic Numbers.

Implementation remarks. The congruence $\alpha g_2 - g_1 \equiv 0 \pmod{2^{k+1}}$ may be checked without performing the full multiplication at each stage, by saving and updating the previous values of $\alpha g_2 - g_1$ and $\alpha f_2 - f_1$. Inside the loop, in the second and third cases, the number d is chosen so as to minimize $\Phi(f + xg)$ (respectively, $\Phi(g + xf)$) among all possible odd integers x. As observed in [47], it may be computed by division. For example, suppose we are in the second case: $\alpha g_2 - g_1 \not\equiv 0 \pmod{2^{k+1}}$ and $\Phi(g) < \Phi(f)$. If $g_1 \neq \pm g_2$, then d is among the odd integers immediately less than or greater than $(f_1 - f_2)/(g_1 - g_2)$ and $-(f_1 + f_2)/(g_1 + g_2)$. Thus it suffices to consider the value of $\Phi(f + dg)$ for these four values of d. When $g_1 = \pm g_2$, one or the other of these quotients is not considered. If $\Phi(g) > \Phi(f)$ then the roles of f and g are switched.

Theorem 10.1 *Let $g = (g_1, g_2)$ denote the output of the preceding algorithm when T bits a_i are used. Then g_2 is odd,*

$$\alpha \cdot g_2 - g_1 \equiv 0 \pmod{2^T},$$

and any other pair $g' = (g'_1, g'_2)$ which satisfies these two conditions has $\Phi(g') \geq \Phi(g)$.

Theorem 10.2 *Suppose $\mathbf{a} = (a_0, a_1, a_2, \cdots)$ is an eventually periodic sequence with associated 2-adic number $\alpha = \sum a_i 2^i = p/q$, with $p, q \in \mathbf{Z}$, and $\gcd(p, q) = 1$. If $T \geq \lceil 2\phi_2(\mathbf{a}) \rceil + 2$ bits a_i are used, then the 2-adic Rational Approximation algorithm outputs $g = (p, q)$. (Hence also if $T \geq 2\lambda_2(\mathbf{a}) + 2 \lceil \log_2(\lambda_2(\mathbf{a})) \rceil$.)*

The proofs of these two optimality results occupy the rest of this section, and utilize the methods of [32, 47]. Consider the k^{th} *approximation lattice* for the 2-adic number α,

$$L_k = \{h \in \mathbf{Z} \times \mathbf{Z} : \alpha \cdot h_2 - h_1 \equiv 0 \pmod{2^k}\}.$$

Then $L_k \supset L_{k+1} \supset \cdots$. If $f = (f_1, f_2) \in L_k$ then $2f = (2f_1, 2f_2) \in L_{k+1}$. The elements $(f_1, f_2) \in L_k$ with f_2 odd represent fractions f_1/f_2 whose 2-adic expansion agrees with that of α in the first k places. Two pairs of integers $f, g \in L_k$ form a *basis* for L_k if every element $h \in L_k$ may be written $h = cf + dg$ for some integers $c, d \in \mathbf{Z}$. Such bases exist and are described in the following lemma, which is a key observation of [47]. Its proof is straightforward:

Lemma 10.3 *Two pairs of integers $f, g \in L_k$ form a basis for L_k if and only if $|f_1 g_2 - f_2 g_1| = 2^k$.*

Proof. Suppose $|f_1 g_2 - f_2 g_1| = 2^k$. Let $h = (h_1, h_2) \in L_k$. We search for $c, d \in \mathbf{Z}$ so that

$$\begin{aligned} cf_1 + dg_1 &= h_1 \\ cf_2 + dg_2 &= h_2. \end{aligned}$$

By Cramer's rule, $c = (h_1 g_2 - g_1 h_2)/(f_1 g_2 - g_1 f_2)$ while $d = (f_1 h_2 - h_1 f_2)/(f_1 g_2 - g_1 f_2)$. Since $f, g, h \in L_k$ the numerators of these expressions are $\equiv 0 \pmod{2^k}$, so they are multiples of 2^k. It follows that c and d are integers, and we conclude that f, g form a basis of L_k, which proves the "if" part. In particular, $f' = (2^k, 0)$ and $g' = (\alpha_k, 1)$ form a basis of L_k (where $\alpha_k = \alpha_0 + \alpha_1 2 + \ldots + \alpha_{k-1} 2^{k-1}$). Next, let us suppose that f, g is any other basis of L_k. Then f and g may be written as integer linear combinations of f' and g', i.e., there is a two by two matrix C of integers so that

$$\begin{pmatrix} f_1 & g_1 \\ f_2 & g_2 \end{pmatrix} = C \cdot \begin{pmatrix} f'_1 & g'_1 \\ f'_2 & g'_2 \end{pmatrix}$$

On the other hand, it is possible to write f' and g' as integer linear combinations of f and g using another two by two integer matrix C'. It follows that $C' = C^{-1}$. So both $\det(C)$ and $\det(C^{-1}) = \det(C)^{-1}$ are integers. Hence $\det(C) = \pm 1$. Therefore $f_1 g_2 - g_1 f_2 = \pm(f'_1 g'_2 - g'_1 f'_2) = \pm 2^k$. $\square$

The proof of Theorem 10.1 is an immediate consequence of the following lemma.

Lemma 10.4 *For each k, at the top of the loop the following conditions hold:*

1. *f and g are in L_k; f_1 and f_2 are even; g_2 is odd;*
2. *$\langle f, g\rangle$ is a basis for L_k;*
3. *$f \notin L_{k+1}$;*
4. *g minimizes $\Phi(h)$ over all elements $h \in L_k$ with h_2 odd;*
5. *f minimizes $\Phi(h)$ over all elements $h \in L_k$ with h_1 and h_2 even.*

Proof: We prove this by induction on k. It is straightforward to check that the conditions (1)-(5) hold initially. Let us suppose that the conditions hold at stage k. If $g \in L_{k+1}$, then after passing through the loop and returning to the top, the new values of f and g are $f' = 2f$ and $g' = g$, and it is straightforward to verify that the conditions hold. Therefore, assume we are back at stage k, and that $g \notin L_{k+1}$. Let f' and g' be the new values after updating. We treat the case when $\Phi(g) < \Phi(f)$. The other case is similar.

1. We have

$$\begin{aligned}\alpha \cdot g_2' - g_1' &= \alpha \cdot (f_2 + dg_2) - (f_1 + dg_1)\\ &= (\alpha \cdot f_2 - f_1) + d(\alpha \cdot g_2 - g_1)\\ &\equiv 2^k + d2^k \pmod{2^{k+1}}\\ &\equiv 0 \pmod{2^{k+1}},\end{aligned}$$

since f and g are in $L_k - L_{k+1}$ and d is odd. Therefore $g' \in L_{k+1}$. Also, g is in L_k, so $f' = 2g$ is in L_{k+1}. The parity conditions on f and g are straightforward to check.

2. By Lemma 10.3, we have $f_1g_2 - f_2g_1 = 2^k$. Therefore $f_1'g_2' - f_2'g_1' = 2g_1(f_2 + dg_2) - 2g_2(f_1 + dg_1) = 2(f_1g_2 - f_2g_1) = 2^{k+1}$. Again by Lemma 10.3, $\langle g', f'\rangle$ is a basis for L_{k+1}.

3. We have $g \notin L_{k+1}$, so $f' = 2g \notin L_{k+2}$.

4. Suppose that minimality fails. Since $\langle f', g'\rangle$ form a basis for L_{k+1}, there are integers a and b so that

$$\Phi(ag' + bf') < \Phi(g') \tag{21}$$

and $ag_2' + bf_2'$ is odd. The latter condition is equivalent to a being odd since f_2' is even and g_2' is odd. By possibly negating both a and b, we can assume a is nonnegative. Further, if $a = 1$, then $ag' + bf' = f + (d + 2b)g$ and this contradicts the choice of d in the algorithm. Thus we can assume that $a > 1$. Equation (21) can be rewritten

$$\Phi(af + (ad + 2b)g) < \Phi(f + dg). \tag{22}$$

Let c be the odd integer closest to $d + 2b/a$. Since a is odd, the quantity $x = c - (d + 2b/a)$ satisfies $|x| < 1$ hence $|x| \le (a-1)/a$. Then

$$\begin{aligned}\Phi(af + acg) &= \Phi(af + (ad + 2b + ax)g) \\ &\le \Phi(af + (ad + 2b)g) + a|x|\Phi(g)\end{aligned}$$

by the triangle inequality for Φ. The first term may be bounded using (22) and the second term is bounded by the induction hypothesis: $\Phi(g) \le \Phi(f + dg)$. Dividing by a gives

$$\Phi(f + cg) < \frac{1}{a}\Phi(f + dg) + |x|\Phi(f + dg) \le \Phi(f + dg)$$

which contradicts the choice of d.

5. Suppose there is an element $h' \in L_{k+1}$ with h'_1 and h'_2 even, such that $\Phi(h') < \Phi(f') = 2\Phi(g)$. We can write $h' = 2h$ for some $h \in L_k$, so $\Phi(h) < \Phi(g) < \Phi(f)$. If both h_1 and h_2 are even, this contradicts the minimality of f. If h_2 is odd, this contradicts the minimality of g. It is impossible that h_1 is odd and h_2 is even for $k \ge 1$ since $\alpha \cdot h_2 - h_1 \equiv 0 \bmod 2^k$. □

Remarks. The algorithm runs correctly if we always update g and f by the first method ($\langle g, f\rangle = \langle f + dg, 2f\rangle$), independent of the relation between $\Phi(g)$ and $\Phi(f)$. The relation $\Phi(g) < \Phi(f)$ was only used to verify property (5) above, which is not necessary for rapid convergence of the algorithm. However, property (5) ensures that the size of f remains small, so it leads to better bounds on the complexity of the computations which are involved in the algorithm. Since the algorithm is adaptive there is, of course, no need to assume that the sequence $\mathbf{a}$ is eventually periodic.

Proof of Theorem 10.2. By assumption, $\alpha = p/q$ so q is odd and $(p, q) \in L_k$ for all k. The output from the algorithm is a pair $g = (g_1, g_2) \in L_T$ which is Φ-minimal, so $\Phi(g_1, g_2) \le \Phi(p, q)$. Hence

$$|g_1 q| \le |g_1||q| \le \Phi(g_1, g_2) \cdot \Phi(p, q) \le \Phi(p, q)^2 \le 2^{T-2},$$

since by assumption $T \ge 2\log_2 \Phi(p, q) + 2$. Similarly, $|pg_2| \le 2^{T-2}$. However, $\alpha g_2 - g_1 \equiv 0 \pmod{2^T}$ so $g_1 q \equiv pg_2 \pmod{2^T}$, which implies that $g_1 q = pg_2$. Therefore (g_1, g_2) is some odd integer multiple of (p, q). By Φ-minimality, this integer must be ± 1 which gives $g_1 = p$ and $g_2 = q$ (or else $g_1 = -p$ and $g_2 = -q$). □

Complexity Issues Suppose the rational approximation algorithm is executed with a sequence $\mathbf{a}$ which is eventually periodic, with rational associated 2-adic number $\alpha = p/q$. Then the rational approximation algorithm takes $T = 2\log(\Phi(p, q)) + 2 \le 2\lambda_2(\mathbf{a}) + 2\lceil \log(\lambda_2(\mathbf{a}))\rceil + 2$ steps to converge.

Consider the kth step. If $\alpha g_2 - g_1 \not\equiv 0 \bmod 2^{k+1}$, then we say that a discrepancy has occurred. The complexity of the algorithm depends on the number of discrepancies. To simplify the computation of αg_2, we maintain αf_2 as well. When no discrepancy occurs, these values and the value of f can be updated with k bit operations.

Suppose a discrepancy occurs. The minimization step can be done with two divisions of k bit integers. The remaining steps take time $\mathcal{O}(k)$. Then αg_2 and αf_2 can be updated with $\mathcal{O}(k)$ bit operations and two multiplications of k bit integers by d.

Let D be the number of discrepancies, and let M be the maximum time taken by a multiplication or division of T bit integers. The Schönhage-Strassen algorithm [46], gives $M = \mathcal{O}(T \log T \log\log T)$. This can be improved to $M \sim T \log T$ using Pollard's nonasymptotic algorithm and Newton interpolation for $T < 2^{37}$ on a 32 bit machine or $T < 2^{70}$ on a 64 bit machine [42]. These are ranges that are typical in current usage.

The complexity of the algorithm is thus $4DM + \mathcal{O}(T^2)$. Strictly in terms of T, this is $\mathcal{O}(T^2 \log T \log\log T)$. However, if the sequence is chosen so that the number of discrepancies is small, the complexity is lower. In particular a cryptographer designing a stream cipher should try to choose sequences for which many discrepancies occur.

11 The Summation Cipher

In the summation cipher [37, 44], several m-sequences $\mathbf{a_1}, \mathbf{a_2}, \cdots, \mathbf{a_k}$ are combined using "addition with carry". The resulting sequence is used as a pseudo-one-time-pad. These sequences have generated great interest since they appear to be resistant to attacks based on the Berlekamp-Massey algorithm. If the constituent sequences $\mathbf{a_i}$ have coprime periods T_i then the resulting sequence has linear span which is close to the period $L = T_1 \cdot T_2 \cdots T_k$ of the combined sequence.

However, by Theorem 9.6 and a generalization of Theorem 9.5, the 2-adic complexity of the combined sequence is no more than $T_1 + T_2 + \cdots + T_k + \log_2(k)$ so the 2-adic span is no more than $\sum T_i + \log_2(k) + \log_2(\sum T_i + \log_2(k))$. Thus if the T_i are similar in magnitude, the 2-adic span of the result is bounded by $kL^{1/k} + \log_2(k) + \log_2(kL^{1/k} + \log_2(k))$ and it may be much less. This throws considerable doubt on the security of these stream ciphers.

Here is a more algorithmic description of the attack:

D1. Determine (perhaps by a known plaintext attack on a stream cipher system) $2 \cdot T$ consecutive bits of the sequence $\mathbf{b}$ formed by applying the summation combiner to $\mathbf{a_1}, \mathbf{a_2}, \cdots, \mathbf{a_k}$.

D2. Apply the rational approximation algorithm to this sequence of bits, to find q and p.

D3. Construct the FCSR which outputs the bit stream corresponding to the 2-adic number $\alpha = p/q$ using the methods in Sections 4 and 5.

The resulting FCSR uses at most $T = T_1 + T_2 + \cdots + T_k + \mathcal{O}(\log(\sum_i (T_i))) + \mathcal{O}(\log(k))$ bits of storage and outputs the sequence $\mathbf{b}$. The effectiveness of this attack may be minimized by using only $k = 2$ constituent m-sequences, with periods T_i chosen so that $2^{T_i} - 1$ has no small prime factors, but then there is less benefit in the growth of the linear span (cf. [38, 39]).

For example, if we combine m-sequences of period $2^L - 1$ with $L = 7$, 11, 13, 15, 16, and 17, then the period and linear span of the resulting sequence are nearly 2^{79}. However, the 2-adic span is less than 2^{18}, so fewer than 2^{19} bits suffice to find a FCSR that generates the sequence. The maximum number of single word arithmetic operations performed by the rational approximation algorithm on a 32 bit machine is about 2^{42}.

12 Relationship to Arithmetic Codes

Fix an odd, positive integer A. Recall that the codewords of the AN (arithmetic) code consist of the binary representations of the integers AN, where the integers N are restricted to lie in some suitable range [41, 43].

Suppose $q > 0$ is an odd integer which we use as the connection integer for a FCSR. Set $T = \mathrm{ord}_q(2)$. Then the output sequence of a FCSR with connection integer q is periodic and the period divides T. If the value of a given state of the FCSR is $\alpha = -p/q$ and if this fraction is in lowest terms, then the period is exactly T.

Theorem 12.1 *The single periods of the periodic sequences generated by the FCSR are precisely the codewords for the cyclic AN code where $A = (2^T - 1)/q$ and $0 \le N < q$.*

Proof: If the output sequence $(a_0, a_1, \cdots)$ is periodic, then by equation (3) the 2-adic value of the shift register is

$$\alpha = -p/q = -\frac{\sum_{i=0}^{T-1} a_i 2^i}{2^T - 1}.$$

Therefore,

$$\sum_{i=0}^{T-1} a_i 2^i = p \cdot \frac{(2^T - 1)}{q} = p \cdot A,$$

which shows that the first T bits are precisely the binary expansion of the integer pA. $\square$

13 Long Sequences

It is desirable to generate pseudorandom sequences with large periods using simple shift register hardware. In the case of linear feedback shift registers, sequences of maximal period are obtained by using a primitive connection polynomial. By Corollary 2.2, the maximum possible period for a FCSR with connection integer q is $T = q - 1$. This period is attained if and only if q is prime and 2 is a primitive root modulo q. In this case, for any initial loading of the register, the output sequence will either degenerate into all 0's or all 1's, or else it will eventually drop into the big periodic state (see Section 5). To emphasize the analogy with m-sequences, we make the following definition.

Definition 13.1 *An ℓ-sequence is a periodic sequence (of period $T = q - 1$) which is obtained from a FCSR with prime connection integer q for which 2 is a primitive root.*

By Theorem 2.1 and Corollary 2.2, such a sequence is (a shift of) the reverse of the binary expansion,

$$1/q = b_0 2^{-1} + b_1 2^{-2} + b_3 2^{-3} + \cdots$$

of the fraction $1/q$ (see, for example, [29] Section 4.1, ex. 31). This binary expansion is called a $1/q$-*sequence* in [3], any single period of which is a codeword in the Barrows-Mandelbaum arithmetic code [2, 33]. These sequences are balanced [12] and they have the generalized de Bruijn property [33, 3, Theorem 1, p. 370]: in any given period of the sequence, every binary string of length $\lfloor \log_2(q) \rfloor$ occurs at least once and every binary string of length $\lfloor \log_2(q) \rfloor + 1$ occurs at most once. The generation of Barrows-Mandelbaum arithmetic codes using FCSR circuitry is new.

The autocorrelation function of a periodic binary ℓ-sequence $\mathbf{a}$ is in general quite difficult to determine. However, there is a well behaved autocorrelation function "with end-around carry" [33], $R_i(\mathbf{a})$, which is an appropriate arithmetic analog to the usual autocorrelation function. (If $\alpha = \sum a_n 2^n$ and $\alpha[i] = \sum a_n 2^{n+i} = 2^i \alpha$ denotes the 2-adic numbers corresponding to the sequence $\mathbf{a}$ and to its shift by i steps, then $R_i(\mathbf{a})$ is the number of 1's minus the number of 0's in any period of the periodic tail of the bit sequence for the sum $\alpha + \alpha[i]$.) Mandelbaum proved that (for $0 \le i < q - 1$) this function is 2-valued with $R_i(\mathbf{a}) = 0$ unless $i = (q-1)/2$, in which case $R_i(\mathbf{a}) = 1$. (See also [27].)

Length	Values of q giving ℓ-sequences
1	3
2	5
3	11, 13
4	19, 29
5	37, 53, 59, 61
6	67, 83, 101, 107
7	131, 139, 149, 163, 173, 179, 181, 197, 211, 227
8	269, 293, 317, 347, 349, 373, 379, 389, 419, 421, 443, 461, 467, 491, 509

Table 2: Values of q Giving Rise to ℓ-sequences for Length ≤ 8.

There are efficient techniques for finding large primes q for which 2 is a primitive root (see [8]) which are already implemented in current software systems such as Maple and Pari. For example, a FCSR based on the prime number

$$q = 2^{128} + 2^5 + 2^4 + 2^2 - 1$$

needs only 2 bits of memory and has maximal period $T = q - 1$. Heilbronn (revising Artin's conjecture) conjectured, and Hooley [17] proved, that if an extension of the Riemann hypothesis to the Dedekind zeta function over certain Galois fields is true, then the number $N(n)$ of primes $q < n$ for which $\mathrm{ord}_q(2) = q - 1$ is

$$N(n) = A \cdot \frac{n}{\ln_2(n)} + O\left(\frac{n \ln_2 \ln_2(n)}{\ln_2^2(n)}\right),$$

where A (= .3739558136 to ten decimals) is Artin's constant. In other words, 37.4% of all prime numbers have this property.

There is another case in which large periods can be obtained. Suppose that q is a prime number such that $q = 2p + 1$ with p a prime number. Then sequences generated by a FCSR with connection integer q will have period $T \geq (q-1)/2$, which is half the maximum possible period. (This is because Fermat's congruence states that, if x is not a multiple of q, then $x^{q-1} \equiv 1 \pmod{q}$, so $\mathrm{ord}_q(2)$ divides $q - 1 = 2p$ and hence is equal either to 2, which is impossible; to p; or to $q - 1$.) It is apparently easier to check whether $(q-1)/2$ is prime than it is to determine whether 2 is a primitive root modulo q. It was conjectured by Hardy and Littlewood [15], and is widely believed by number theorists, that the number of primes $P(n)$ less than n of the form $2p + 1$, p prime, is asymptotically given by

$$P(n) \sim c_2 \cdot \frac{n}{\ln^2(n)},$$

where c_2 (= .0330080908 to ten decimal places) is a constant.

Finally, long pseudorandom sequences may be generated by FCSR's with nonprime connection integer q. By theorem 6.1 a large period is obtained if the powers of 2 constitute a large cyclic subgroup of $(\mathbf{Z}/(q))^*$ (the collection of invertible elements in $\mathbf{Z}/(q)$). The order of $(\mathbf{Z}/(q))^*$ is given by Euler's phi function, $\phi(q)$ (the number of integers less than q which are relatively prime to q). The group $(\mathbf{Z}/(q))^*$ is cyclic if and only if q is a power of a prime, say $q = p^e$. To determine whether 2 is primitive modulo p^e, it suffices to consider primitivity modulo p and p^2 [18, §4.1 Theorem 2].

Proposition 13.2 *Suppose $q = p^e$ with p an odd prime and $e \geq 2$. Then 2 is primitive modulo q if and only if 2 is primitive modulo p^2. This holds if and only if 2 is primitive modulo p and p^2 does not divide $2^{p-1} - 1$. In this case the period is $\phi(p^e) = p^{e-1}(p-1) = q(p-1)/p$.*

One can ask about the abundance of primes p for which 2 is a primitive root modulo p^2. All of the primes p listed in table 2 have this property. In fact Hardy and Wright point out that the condition that p^2 divides $2^{p-1} - 1$ holds for only two primes p less than $3 \cdot 10^7$ [16, p. 73], and by computer search E. Bombieri has extended this limit to $2 \cdot 10^{10}$ [5]. (The two primes are 1093 and 3511.) In both cases 2 is not primitive modulo p. Thus for a large number of primes, we need only check the primitivity of 2 modulo p. In fact, it is not known whether there are *any* primes p such that 2 is primitive modulo p but not modulo p^2, though there is no compelling reason to believe there are no such primes.

Distributional Properties of ℓ-Sequences

We next show that the sequences based on prime power connection integers for which 2 is a primitive root have excellent distributional properties. That they are balanced follows easily from the primitivity of 2.

Proposition 13.3 *Let q be a power of a prime p, say $q = p^e$, and suppose that 2 is primitive modulo q. Let* **a** *be any maximal period FCSR sequence, generated by a FCSR with connection integer q. The number of zeros and the number of ones in one period of* **a** *are equal.*

Furthermore we can consider higher order distributions. We show next that these sequences are close to having the deBruijn property that each subsequence of length log of the period occurs exactly once in each period. We show that for any two such subsequences, their numbers of occurences can differ by at most two.

Theorem 13.4 *Let q be a power of a prime p, say $q = p^e$, and suppose that 2 is primitive modulo q. Let s be any nonnegative integer, and let A and B be s bit subsequences. Let* **a** *be any maximal period, purely periodic FCSR sequence, generated by a FCSR with connection integer q. Then the numbers of occurrences of A and B in* **a** *with their starting positions in a fixed period of* **a** *differ by at most 2.*

Proof: The purely periodic FCSR sequences with connection integer q are precisely the 2-adic expansions of rational numbers $-x/q$, with $0 \le x < q$. Such a sequence has maximum period if and only if p does not divide x. Since 2 is primitive modulo q, the cyclic shifts of $\mathbf{a}$ correspond to the set of all rational numbers $-x/q$, with $0 \le x < q$. Thus an s bit subsequence A occurs in $\mathbf{a}$ if and only if it occurs as the first s bits in the 2-adic expansion of some rational number $-x/q$ with $0 \le x < q$ and p not dividing x. Two rational numbers $-x_1/q$ and $-x_2/q$ have the same first s bits if and only if $-x_1/q \equiv -x_2/q \bmod 2^s$, i.e., if and only if $x_1 \equiv x_2 \bmod 2^s$. Thus we want to count the number of x with a given first s bits, $0 \le x < q$, and x not divisible by p.

Let $2^r < q < 2^{r+1}$. If $s > r$, there are either zero or one such x, so the result follows. Thus we may assume $s \le r$.

We first count the number of x with the first s bits fixed and $0 \le x < q$, ignoring the divisibility condition. If $A = a_0, \cdots, a_{s-1}$, we let $\alpha = \sum_{i=0}^{s-1} a_i 2^i$. Let $q = \sum_{i=0}^{r} q_i 2^i$, and $q' = \sum_{i=0}^{s-1} q_i 2^i$. If $\alpha < q'$, then every choice of $a_s, \cdots, a_r$ with $\sum_{i=s}^{r} a_i 2^i \le \sum_{i=s}^{r} q_i 2^i$ gives a unique x in the right range. If $\alpha \ge q'$, then every choice of $a_s, \cdots, a_r$ with $\sum_{i=s}^{r} a_i 2^i < \sum_{i=s}^{r} q_i 2^i$ gives a unique x in the right range. Thus for different choices of A, the numbers of such x differ by at most one.

Next we consider those x for which $0 \le x < q$ and p divides x. That is, $x = py$ for some y, and $0 \le y < q/p = p^{e-1}$. As above, $x_1 = py_1$ and $x_2 = py_2$ have the same first s bits if and only if the same is true of y_1 and y_2. The preceding paragraph shows that the numbers of such y for different choices of the first s bits differ by at most one. But if $x = py$, then $y \equiv A \bmod 2^s$ if and only if $x \equiv pA \bmod 2^s$, so for any B and C, the number of xs divisible by p with first s bits equal to B differs from the number of xs divisible by p with first s bits equal to C by at most 1. We have

$$\begin{aligned} &|\{x : 0 \le x < q,\, p \not| x, \text{ and } x \equiv \alpha \bmod 2^s\}| \\ &\quad = \; |\{x : 0 \le x < q \text{ and } x \equiv \alpha \bmod 2^s\}| - |\{x : 0 \le x < q,\, p|x, \text{ and } x \equiv \alpha \bmod 2^s\}|. \end{aligned}$$

As α varies the two terms on the right hand side vary by at most one from their values for any fixed choice of α. Thus the difference varies by at most 2. □

It is easy to check that the difference can be as large as 2.

14 Related Constructions and Open Problems

In this section we hope to convince the reader that there is an endless assortment of variations on the idea of the FCSR, most of which may be analyzed along the lines we have outlined, perhaps by using more sophisticated mathematical tools.

Most of the results in this paper have straightforward generalizations to FCSRs with cell contents and feedback coefficients in $\mathbf{Z}/(p)$ where p is a prime number, not necessarily 2. Let

$$q = -1 + q_1 p + q_2 p^2 + \cdots + q_r p^r$$

denote the base p expansion of a positive integer $q \equiv -1 \pmod p$. Then q is the *connection integer* for a FCSR with feedback coefficients $q_1, q_2, \cdots, q_r$ in $\mathbf{Z}/(p)$ as in Figure 2. With each clock cycle, the integer sum $\sigma_n = \sum_{k=1}^{r} q_k a_{n-k} + m_{n-1}$ is accumulated, the register contents are shifted one cell to the right, the quantity $a_n = \sigma_n \pmod p$ is placed in the leftmost cell, and the new memory value is $m_n = \lfloor \sigma_n / p \rfloor$.

Let $w = \sum_{i=1}^{r} q_i$ denote the sum of the coefficients in the base p expansion of $q+1$. For any initial value of the memory $m \ge 0$, as the register runs, the memory will decrease until it is $\le w$, and then then the memory will remain $\le w$.

Suppose the FCSR is initially loaded with contents $a_{r-1}, \cdots, a_0 \in \mathbf{Z}/(p)$ and with initial memory $m \in \mathbf{Z}$. Then the output of the FCSR is the p-adic expansion of the rational number

$$\alpha = \frac{\sum_{i=0}^{r-1} \sum_{j=0}^{r-i-1} q_i p^i a_j p^j - m p^r}{q} \tag{23}$$

If q is prime, then the periodic part of the output sequence will have period $T = \mathrm{ord}_q(p)$. The output will be strictly periodic if the numerator in equation (23) lies between $-q+1$ and 0. In this case, if $\gamma = p^{-1} \pmod q$ then the successive values output by the FCSR are given by $a_i = A\gamma^i \pmod q \pmod p$ for some constant $A \in \mathbf{Z}/(q)$. If p is primitive modulo q then the output sequence has maximal period $T = q - 1$, is balanced, and has the generalized de Bruijn property.

Some of these observations appear in the important article [36] where linear recurrences with carry are proposed, in the case that $q + 1 = p^a + p^b$ is a sum of two pure powers of p. There, the period of such a sequence is computed and it is observed that, in this case, only one bit of memory is needed.

We may define the *p-adic span* of an eventually periodic pseudorandom sequence $\mathbf{a} = a_0, a_1, a_2, \ldots$ of elements $a_i \in \mathbf{Z}/(p)$ to be the size of the smallest FCSR which generates that sequence, and the *p-adic complexity* of the sequence to be $\log_p(\max(|d|, |q|))$ where

$$\alpha = \frac{d}{q} = \sum_{i=0}^{\infty} a_i p^i$$

is the rational number (in lowest terms) whose p-adic expansion is the sequence $\mathbf{a}$. The p-adic span and p-adic complexity are related as in proposition 9.3 (but with $\log_2$ replaced by $\log_p$). The p-adic complexity of the (p-adic) sum of two sequences is no greater than the sum of their p-adic complexities plus 1.

The algorithm in de Weger [47], which we briefly recall in figure 14, is an efficient way to approximate any finite pseudorandom sequence (with elements in $\mathbf{Z}/(\mathbf{p})$) by a rational number, from which a FCSR may be constructed which duplicates the given sequence. It is not "adaptive": it assumes an input consisting of T elements $a_0, a_1, \ldots, a_{T-1}$ of the sequence. It exhibits an optimality property analogous to that of theorem 10.2, but we do not know whether there is an adaptive algorithm which also exhibits the first optimality property. (Theorem 10.1). (Our analysis in section 10 is strictly specific to base 2.) If $f = (f_1, f_2)$ is a pair of integers, we use the previous notation $\Phi(f) = \max(|f_1|, |f_2|)$.

de Weger$(a_0, a_1, \ldots, a_{T-1})$
    begin
    $\alpha = a_0 + a_1 p + \ldots + a_{T-1} \cdot p^{T-1}$
    $f = (0, p^{T-1})$
    $g = (p^{T-1}, 1)$
    repeat
        Let $d \in \mathbf{Z}$ minimize $\Phi(f + dg)$
        $f = f + dg$
        Interchange f, g
    until $\Phi(g) \leq \Phi(f)$
    return g
    end

Figure 5: De Weger's algorithm for p-adic numbers

It is possible to design, analyze, and build feedback with carry shift registers for which the cells contain elements of some other finite field $GF(p^m)$. We refer the reader to [22] for further information.

A very interesting variant on the FCSR architecture (which we call a d-FCSR) is shown in figure 6. Each cell contains 0 or 1. The operation is analogous to that of the FCSR in figure

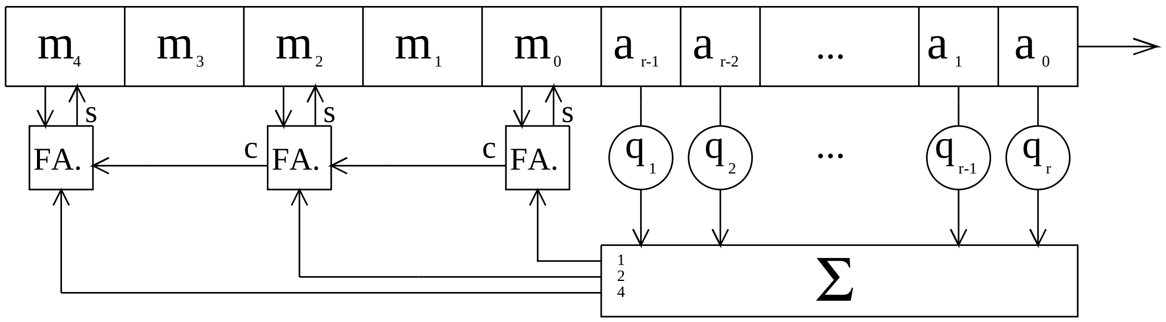

Figure 6: A d-FCSR with $d = 2$.

3 except that each "carried" bit is delayed d steps before being added. There is an analogous d-step "combiner" which works just like the summation combiner [44, Fig. 9.5 p. 217], except that the single cell for memory is replaced by a shift register of length d which delays the carry bit for d clock cycles before adding it back in. We refer to this operation as a sum with d-step carry.

The key mathematical tool for analysis of the d-FCSR is the ring $\mathbf{D}$ of 'π-adic integers". These consist of formal power series $a_0 + a_1\pi + a_2\pi^2 + \cdots$ in an indeterminate π, where $a_i \in \{0, 1\}$ and where π satisfies the formal rule $\pi^d = 2$. Addition and multiplication in this ring are performed just as in the ring $\mathbf{Z}_2$ of 2-adic integers. However, carried bits are advanced d steps because

$$1 + 1 = 0 + 0\pi + \cdots + 0\pi^{d-1} + \pi^d.$$

Thus, the sum with d step carry, described above, is precisely the sum operation in the ring $\mathbf{D}$.

The operations div π and mod π make sense in this ring: If $\sigma = \sigma_0 + \sigma_1\pi + \cdots + \sigma_s\pi^s \in \mathbf{D}$ is a finite sum of powers of π with coefficients $\sigma_i \in \{0, 1\}$, define $\sigma(\bmod \pi) = \sigma_0 \in \mathbf{Z}/(2)$ and $\sigma\ (\text{div } \pi) = \sigma_1 + \sigma_2\pi + \cdots + \sigma_s\pi^{s-1}$. A formal description of a d-FCSR may be given using this language.

A connection "integer" $q = -1 + q_1\pi + q_2\pi^2 + \cdots + q_r\pi^r$ (where $q_i \in \{0, 1\}$) determines taps on a shift register, just as in Figure 2. The contents of the memory m form a polynomial, $m = m_0 + m_1\pi + m_2\pi^2 + \cdots + m_s\pi^s$ with $m_i \in \{0, 1\}$. The register operates as follows:

1. Form the integer sum $\sigma' = \sum_{i=0}^{r-1} a_i q_{r-i}$.

2. Write σ' as a polynomial (with $\{0, 1\}$ coefficients) in π using $2 = \pi^d$.

3. Using addition in $\mathbf{D}$, form the sum $\sigma = m + \sigma' \in \mathbf{D}$.

4. Shift the contents of the register to the right one step.

5. Place the bit $a_r = \sigma \pmod{\pi}$ into the leftmost cell.

6. Replace the memory with $m' = \sigma \ (\text{div } \pi) = (\sigma - a_r)/\pi$.

There are notions of π-adic span and complexity; the π-adic complexity adds when two binary sequences are combined using the d-step carry combiner. Many of the constructions and results in this paper have analogs in this more general setting, which we have described and analyzed briefly in [24], however there remain many interesting open problems concerning these registers.

It is also possible to design and build shift register architectures in which the cells contain elements from some finite field $GF(p^r)$ and in which the addition involves a d-step carry, thus combining all of the above ideas. The appropriate mathematical tool for the analysis of this sort of architecture involves the theory of p-adic fields and their ramified extensions, which goes beyond the scope of the present paper.

There are many relations between FCSR sequences and LFSR sequences which should be studied: what is the linear complexity (profile) of an ℓ-sequence? What is the 2-adic complexity (profile) of an m-sequence? When ℓ-sequences and m-sequences are combined (e.g. by bitwise sum or by summation combiner) does the complexity of the result approach the period? Bruce Schneier's book [45] proposes a number of combiners which should be analyzed for 2-adic and linear complexity. Similar questions may be posed concerning d-FCSR sequences.

For a LFSR with a fixed connection polynomial, the set of output sequences forms a vectorspace; in fact they form the codewords of a first order Reed-Muller code. We do not know any simple characterization of the set of output sequences of a given FCSR.

Adaptive versions of de Weger's algorithm should be developed for other prime bases, and for the d-FCSR architecture. The rate of convergence of de Weger's algorithm (for other prime bases) should be studied. As mentioned in the introduction and in section 9, the appropriate connections with maximum order complexity ([19, 20, 21, 4]) should be made.

References

[1] E. Bach, Efficient prediction of Marsaglia-Zaman random number generators, (draft, Univ. of Wisconsin, 1993).

[2] J. T. Barrows, Jr., A new method for constructing multiple error correcting linear residue codes, Rep. R-277, Coordinated Sci. Lab., Univ. of Illinois, Urbana, 1966.

[3] L. Blum, M. Blum, and M. Shub, A simple unpredictable pseudo-random number generator, *Siam J. Comput.* vol. 15, pp. 364-383 (1986).

[4] A. Blumer and J. Blumer *Linear size finite automata for the set of all subwords of a word: an outline of results* Bulletin of the European Association for Theoretical Computer Science vol. 21, 1983 pp. 68-77.

[5] E. Bombieri, personal communication.

[6] A. Chan and R. Games, On the quadratic span of de Bruijn sequences. *IEEE Trans. Info. Theory* vol. 36, 1990 pp. 822-829.

[7] U. Cheng, On the continued fraction and Berlekamp's algorithm. *IEEE Trans. Info. Theory* vol. 30, 1984 pp. 541-544.

[8] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer Verlag, N. Y., 1993.

[9] Z. D. Dai and K. C. Zeng, Continued fractions and the Berlekamp-Massey algorithm. *Auscrypt '90*, Springer Lecture Notes in Comp. Sci. vol. 453, Springer Verlag, N. Y., 1990.

[10] C. Ding, *Stream Ciphers and Number Theory*, to appear.

[11] H. D. Ebbinghaus et al, *Numbers.* Graduate Texts in Mathematics vol. 123, Springer Verlag, N. Y. (1990).

[12] C. F. Gauss, *Disquisitiones Arithmeticae*, 1801; reprinted in English translation by Yale Univ. Press, New Haven, CT. 1966.

[13] S. Golomb *Shift Register Sequences.* Aegean Park Press, Laguna Hills CA, 1982.

[14] R. T. Gregory and E. V. Krishnamurthy, *Methods and Applications of Error-Free Computation*, Springer Verlag, N. Y., 1984.

[15] G. H. Hardy and J. E. Littlewood, Some problems of 'Partitio Numerorum'; III: on the expression of a number as a sum of primes. *Acta Mathematica* vol. 44, 1922 pp. 1-70.

[16] G. Hardy and E. Wright, *An Introduction to the Theory of Numbers.* Oxford University Press, Oxford UK, 1979.

[17] C. Hooley, On Artin's conjecture. *J. Reine Angew. Math.* vol. 22, 1967 pp. 209-220.

[18] K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, Springer Verlag, N.Y., 1990.

[19] C. J. A. Jansen, *Information theory of shift registers* in *Proceedings of the Tenth Symposium on Information Theory in the Benelux*, A. M. Barbe, ed., Werkgemeenschap voor Inf.- & Communicatietheorie, Enschede, Netherlands, 1989, pp. 153 -160.

[20] C. J. A. Jansen and D. E. Boekee, *The shortest feedback shift register that can generate a given sequence* in *Advances in Cryptology - CRYPTO '89. Proceedings* G. Brassard, ed. Lecture Notes in Computer Science vol. 453, Springer-Verlag, Berlin, 1990, pp. 90-99.

[21] C. J. A. Jansen and D. E. Boekee *On the significance of the directed acyclic word graph in cryptology* in *Advances in Cryptology-AUSCRYPT '90 International Conference on Cryptology. Proceedings* Springer Verlag, Berlin, 1990, pp. 318-326.

[22] A. Klapper, Feedback with carry shift registers over finite fields. *Fast Software Encryption. Second Internatinoal Workshop.* (Lecture Notes in Computer Science, Springer Verlag, N. Y., 1995) pp. 170-178.

[23] A. Klapper and M. Goresky, 2-adic shift registers. *Fast Software Encryption.* (Lecture Notes in Computer Science vol **809**, Springer Verlag, N. Y., 1994.) pp. 174-178.

[24] A. Klapper and M. Goresky, Feedback registers based on ramified extensions of the 2-adic numbers. *Advances in Cryptology - Eurocrypt 1994*, Perugia Italy. (Lecture Notes in Computer Science vol **950**, Springer Verlag, N. Y., 1995.) pp. 215-222.

[25] A. Klapper and M. Goresky, Large period nearly deBruijn FCSR sequences. *Advances in Cryptology - Eurocrypt 1995.* (Springer Lecture Notes in Computer Science vol. **921**, Springer Verlag, N. Y., 1995.) pp. 263-273.

[26] A. Klapper and M. Goresky, Cryptanalysis based on 2-adic rational approximation. *Advances in Cryptology - CRYPTO 95.* (Springer Lecture Notes in Computer Science vol. **963**, Springer Verlag, N. Y., 1995.) pp. 262-273.

[27] A. Klapper and M. Goresky, Arithmetic cross-correlation of FCSR sequences. to appear.

[28] E. V. Krishnamurthy and R. T. Gregory, Mapping integers and Hensel codes onto Farey fractions. *BIT* vol. 23, 1983 pp. 9-20.

[29] D. Knuth, *The Art of Computer Programming, Vol 2. Seminumerical Algorithms.* Addison-Wesley, Reading MA, 1981.

[30] N. Koblitz, *p-Adic Numbers, p-Adic Analysis, and Zeta Functions.* Graduate Texts in Mathematics Vol. 58, Springer Verlag, N. Y. 1984.

[31] A. Lempel, M. Cohn, and W. Eastman, A class of balanced binary sequences with optimal autocorrelation properties, *IEEE Trans. Info. Theory,* vol. IT-23, 1977 pp. 38-42.

[32] K. Mahler, On a geometrical representation of p-adic numbers, *Ann. of Math.* vol. 41, 1940 pp. 8-56.

[33] D. Mandelbaum, Arithmetic codes with large distance. *IEEE Trans. Info. Theory,* vol. IT-13, 1967 pp. 237-242.

[34] D. Mandelbaum, An approach to an arithmetic analog of Berlekamp's algorithm. *IEEE Trans. Info. Theory,* vol. IT-30, 1984 pp. 758-762.

[35] G. Marsaglia, The mathematics of random number generators, *The Unreasonable Effectiveness of Number Theory*, Amer. Math. Soc., Providence R. I., 1992. pp. 73-90.

[36] G. Marsaglia and A. Zaman, A new class of random number generators, *Annals of Applied Probability.* vol. 1, 1991 pp. 462-480.

[37] J. Massey and R. Rueppel, Method of, and Apparatus for, Transforming a Digital Data Sequence into an Encoded Form, U.S. Patent No. 4,797,922, 1989.

[38] sc W. Meier and O. Staffelbach, Correlation properties of combiners with memory in stream ciphers, *Advances in Cryptology - EUROCRYPT '90. Workshop on the Theory and Application of Cryptographic Techniques Proceedings* Springer Verlag, Berlin, 1991, pp. 204-13

[39] sc W. Meier and O. Staffelbach, Correlation properties of combiners with memory in stream ciphers, *Journal of Cryptology* vol. 5, 1992, pp. 67-86

[40] W. H. Mills, Continued fractions and linear recurrences, *Math. Comp.* vol. 29, 1975, pp. 173-180

[41] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes* second edition, MIT Press, Cambridge MA, 1972.

[42] J. Pollard The Fast Fourier Transform in a Finite Field, *Math. Comp.,* vol. 25, 1971, pp. 365-374.

[43] T. R. N. Rao, *Error Coding For Arithmetic Processors*, Academic Press, New York N. Y., 1974

[44] R. Rueppel, *Analysis and Design of Stream Ciphers.* Springer Verlag, New York, 1986.

[45] B. Schneier, *Applied Cryptography.* John Wiley & Sons, New York, 1996.

[46] A. Schönhage and V. Strassen, Schnelle Multiplikation Grosser Zahlen, *Computing,* vol. 7, 1971, pp. 281-292.

[47] B. M. M. de Weger *Approximation lattices of p-adic numbers. J. Num. Th.* vol. 24, 1986, pp. 70-88.

[48] L. R. Welch and R. A. Scholtz, Continued fractions and Berlekamp's algorithm. *IEEE Trans. Info. Theory* vol. 25, 1979 pp. 19-27.