Maximum Length Linear Feedback Shift Registers

(c) Peter Fischer

Institute for Computer Engineering (ZITI)
Heidelberg University, Germany
email address: peter.fischer@ziti.uni-heidelberg.de

February 23, 2018

Linear Feedback Shift Registers (LFSRs) are commonly used in digital circuit design to generate long ‘random’ sequences of 1s and 0s with little hardware effort. In this text I will show how the period of such a sequence obtained in a LFSR with exclusive-or feedback can be calculated. In particular, the conditions to obtain the maximum possible period of $2^N - 1$ for a register of N bit length will be clarified. I recapitulate most of the requited mathematics, so that the text should be understandable with little background.

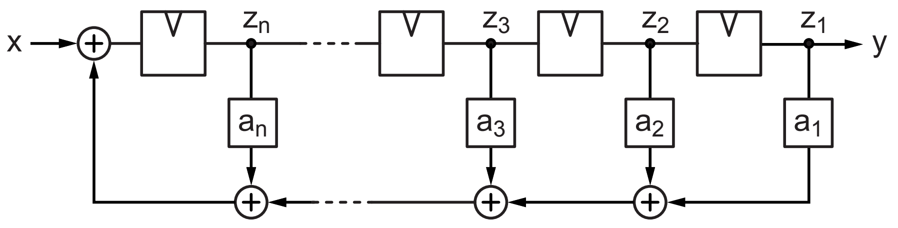

Figure 1: ‘Fibonacci’ type linear shift register with exclusive-or feedback and input signal. The circles with ’+’ signs denote exclusive-or gates. The $a_i \in [0, 1]$ are parameters which affect the properties of the circuit.

1 Introduction

When a digital shift register of N bit length (fig. 1) is fed (at its input) with an exclusive-or combination of the output signal (z_1) and some of the internal bits ($z_2 \dots z_N$), sequences with very interesting properties can be generated. One particularly interesting case is when the shift register cycles through all possible bit combinations, which gives a sequence of length $2^N - 1$ (one less than the total number of bit combinations, because the state with all zeros is stable). Such sequences have very interesting properties so that they are commonly used as noise generators, pseudo random number generators, counters or binary test pattern generators. In order to obtain such a *maximum length sequence*, the internal bits used in the exclusive-or must be at the correct positions. In this text, I want to show how the taps can be found for a given register length N.

The intention is to really go through all the steps in the derivation and to make it understandable for a non mathematics expert. The derivation summarizes the approach presented in [1] and in the standard book on shift registers from Solomon W. Golomb [2]. The following text requires very little mathematical knowledge, just a bit of matrix algebra and no fear of sum notations, indices and so on. Many expressions could be written down in a much more general manner, but for simplicity, I restrict everything to the simplest case.
The derivation will start with a formalization of what is going on in the shift register. We will mix in an external input signal also (in addition to the exclusive-or of an arbitrary combination of shift register bits) to be able to excite the initially empty shift register at time $t = 0$. We will derive a general expression for the output state after an arbitrary number of clocks. We will then introduce the well known *z-transformation* and use it to extract periodicity information from the output sequence. At the end, we will find criteria which must be met by the exclusive-or pattern to yield an output sequence of a given period.

2 Calculation of the Output Signal

Shift Register Description. We consider a shift register with N bit length. The state of the shift register flip flops is described by the N numbers $z_i(t) \in [0, 1]$ with $i = 1 \ldots \mathrm{N}$. The integer time argument $t = 0, 1, \ldots \infty$ tells us how many clocks have elapsed. z_1 is the output of the shift register, while z_N is the value of the first flip flop in the chain. We denote the output also as $y(t) := z_1(t)$ (see fig. 1). For compact notation, we collect the $z_i(t)$ in a (column) vector

$$\vec{z}(t) := \begin{pmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_n(t) \end{pmatrix}. \tag{1}$$

We start with an empty shift register at time $t = 0$:

$$\vec{z}(0) = \vec{0}. \tag{2}$$

Feedback and Input Signal. Some internal signals of the register which we denote as *taps* and an additional input signal $x(t)$ are exor-ed together. This signal forms the input to the first flip flop (see fig. 1). For a general treatment, we multiply every tap z_i by a coefficient $a_i \in [0, 1]$ $(i = 1 \ldots \mathrm{N})$ and sum up all products plus the external input x. When we do the summing modulo 2 (i.e. 1+1 = 0, or '+'='-'), this sum is just an exclusive or. (Mathematicians would say that we are working on GF(2), the *Galois Field* with two elements.) Note that we obviously must always use the tap at the end of the register (i.e. we must have $a_1 = 1$), as we would not really use the N bits otherwise. For our purpose, we can restrict the input signal $x(t)$ to a single excitation at $t = 0$, i.e.

we have a *delta pulse* at the input:

$$x(t) = x_\delta(t) := \delta_{t,0} = \begin{cases} 1 & \text{for} \quad t = 0 \\ 0 & \text{for} \quad t > 0 \end{cases} . \tag{3}$$

General Output Signal. For the output $y(t)$ and the state bits $z_i(t)$ we obviously have

$$\begin{aligned} z_i(t+1) &= z_{i+1}(t) \quad \text{for} \quad i = 1 \ldots \mathrm{N}-1 \\ z_\mathrm{N}(t+1) &= x(t) + \sum_{i=1}^{\mathrm{N}} a_i \cdot z_i(t) \\ y(t) &= z_1(t). \end{aligned}$$

The sum is calculated modulo 2, of course. These equations can be written very nicely using a matrix notation as

$$\vec{z}(t+1) = \mathbf{A}\,\vec{z}(t) + \mathbf{B}\,x(t) \tag{4}$$
$$y(t) = \mathbf{C}\,\vec{z}(t). \tag{5}$$

A, **B** and **C** have dimensions of $\mathrm{N} \times \mathrm{N}, 1 \times \mathrm{N}$ and $\mathrm{N} \times 1$, respectively, and are defined as

$$\mathbf{A} := \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_\mathrm{N} \end{pmatrix}, \mathbf{B} := \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{6}$$

and

$$\mathbf{C} := (1\,0\,0\,\ldots\,0)\,. \tag{7}$$

Output Signal after Time t. We now use (4) to calculate the shift register state for successive time steps starting at $t = 0$ with the initial condition (2) and the properties of the input signal (3):

$$\begin{aligned} \vec{z}(1) &= \mathbf{A}\,\vec{z}(0) + \mathbf{B}\,x(0) = \mathbf{B} \\ \vec{z}(2) &= \mathbf{A}\,\vec{z}(1) + \mathbf{B}\,x(1) = \mathbf{A}\,\mathbf{B} \\ \vec{z}(3) &= \mathbf{A}\,\vec{z}(2) + \mathbf{B}\,x(2) = \mathbf{A}^2\mathbf{B} \\ &\vdots \\ \vec{z}(t) &= \mathbf{A}^{t-1}\,\mathbf{B}. \end{aligned} \tag{8}$$

According to (5), the output signal $y(t)$ is therefore

$$y(t) = \mathbf{C}\,\mathbf{A}^{t-1}\,\mathbf{B}. \tag{9}$$

3 The z-Transformation

For the next steps, we will need the *z-Transformation* $\mathcal{Z}[f]$ of an infinite sequence of discrete function values given by $\{f(t)\} = f(0), f(1), \ldots.$ It is defined as

$$F(s) = \mathcal{Z}[f(t)] := \sum_{j=0}^{\infty} f(j)\, s^j. \tag{10}$$

$F(\ldots)$ depends on a variable which is often named z, we use s for compatibility with other texts. $F(s)$ has several interesting properties which we will derive next.

z-Transform of a Delta Pulse. The z-transform of a delta pulse at $t = 0$ is constant:

$$\mathcal{Z}[x_\delta(t)] = \sum_{j=0}^{\infty} \delta_{t,0}\, s^j = 1. \tag{11}$$

z-Transform of a Time Shifted Signal. When the input sequence is shifted by one time step, the z-transform is basically just divided by s (the extra term often vanishes):

$$\begin{aligned} \mathcal{Z}[f(t+1)] &= \sum_{j=0}^{\infty} f(j+1)\, s^j = \sum_{j=1}^{\infty} f(j)\, s^{j-1} \\ &= \frac{\mathcal{Z}[f(t)] - f(0)}{s}. \end{aligned} \tag{12}$$

z-Transform of a Periodic Signal. Let us assume that the sequence $f(t)$ has a period of P starting at time $t = \tau$, i.e. that the sequence looks like this:

$$\begin{array}{lll} \{f(t)\} = & r_0, r_1, \ldots, r_{\tau-1}, & \text{('preamble', } \tau \text{ values)} \\ & r_\tau, r_{\tau+1}, \ldots, r_{\tau+P-1}, & \text{(first period, P values)} \\ & r_\tau, r_{\tau+1}, \ldots & \text{(next periods...)} \end{array} \tag{13}$$

The z-transform of this periodic signal is given by

$$\begin{aligned} \mathcal{Z}[f(t)] &= r_0 + r_1 s + \ldots + r_{\tau-1} s^{\tau-1} + \\ &\quad r_\tau s^\tau + \ldots + r_{\tau+P-1} s^{\tau+P-1} + \\ &\quad r_\tau s^{\tau+P} + \ldots + r_{\tau+P-1} s^{\tau+2P-1} + \ldots \\ &= \sum_{i=0}^{\tau-1} r_i s^i + s^\tau (1 + s^P + \ldots) \times (r_\tau + \ldots + r_{\tau+P-1} s^{P-1}) \\ &= \sum_{i=0}^{\tau-1} r_i s^i + \frac{s^\tau}{1 + s^P} \sum_{i=0}^{P-1} r_{\tau+i} s^i. \end{aligned} \tag{14}$$

We have simplified the expression by replacing the infinite geometric sum $(1 + s^P + \ldots)$ by $(1 - s^P)^{-1}$ and by switching to a '+' sign in the denominator, because we work modulus 2.

4 z-Transform of the Output Sequence

We can now use the z-Transform to study the periodicity of the output signal $y(t)$ as given by (9). We apply the z-Transform to (4) using the delta pulse input (3):

$$\begin{aligned}
\mathcal{Z}[\vec{z}(t+1)] &= \mathbf{A}\,\mathcal{Z}[\vec{z}(t)] + \mathbf{B}\,\mathcal{Z}[x_\delta(t)] \\
s^{-1}\mathcal{Z}[\vec{z}(t)] &= \mathbf{A}\,\mathcal{Z}[\vec{z}(t)] + \mathbf{B} \\
(\mathbf{I} - s\mathbf{A})\,\mathcal{Z}[\vec{z}(t)] &= s\,\mathbf{B} \\
\mathcal{Z}[\vec{z}(t)] &= s\,(\mathbf{I} - s\mathbf{A})^{-1}\mathbf{B} \qquad (15)
\end{aligned}$$

In the first step, we have used (11) and (12) and we have introduced the $\mathrm{N} \times \mathrm{N}$ unit Matrix $\mathbf{I}$. From (5) we then get

$$\begin{aligned}
\mathcal{Z}[y(t)] &= \mathbf{C}\,Z[\vec{z}(t)] \\
&= s\,\mathbf{C}\,(\mathbf{I} - s\mathbf{A})^{-1}\mathbf{B} \qquad (16)
\end{aligned}$$

This result could have been directly obtained, in principle, by transforming (9). It looks a bit complicated, but because $\mathbf{B}$ and $\mathbf{C}$ both contain just one non-zero element (see (6) and (7)), we in fact only need the upper right element of the inverse matrix in the middle.
What follows now is a bit tedious, and can be skipped up to the result (18). A general expression for the inverse of a matrix $\mathbf{M}$ with elements $M_{i,j}$ is $(\mathbf{M}^{-1})_{i,j} = |\mathbf{M}^\dagger_{j,i}|/|\mathbf{M}|$ where $|\ldots|$ is the determinant and $|\mathbf{M}^\dagger_{i,j}|$ is the adjoint $(\mathrm{N}-1) \times (\mathrm{N}-1)$ matrix. It is obtained by crossing out the i-th column and the j-th row in $\mathbf{M}$, multiplied by an extra sign factor $(-1)^{i+j}$. The matrix $\mathbf{M} := \mathbf{I} - s\mathbf{A}$ in the middle looks explicitly like this:

$$\mathbf{M} = \begin{pmatrix} 1 & -s & 0 & \cdots & 0 \\ 0 & 1 & -s & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & -s \\ -a_1 s & -a_2 s & -a_3 s & \cdots & 1 - a_\mathrm{N} s \end{pmatrix}. \qquad (17)$$

For the calculation of the required $(\mathbf{M}^{-1})_{\mathrm{N},1}$, we need $|\mathbf{M}^\dagger_{1,\mathrm{N}}|$ which we obtain by crossing out the leftmost column and the lowermost row in $\mathbf{M}$. The remaining matrix has $(\mathrm{N}-1)$ times $-s$ on the diagonal and only zeros in the upper right part, so that its determinant is just $(-s)^{\mathrm{N}-1}$. Together with the $(-1)^{\mathrm{N}+1}$ sign factor, this gives just $s^{\mathrm{N}-1}$. One method to calculate the determinant of $\mathbf{M}$ is as follows: go through all elements $M_{i,j}$ in a fixed row j and sum up $(-1)^{i+j} \cdot M_{i,j} \cdot |\mathbf{M}^\dagger_{i,j}|$. Applying this to the bottom row, we get a sign factor $(-1)^{\mathrm{N}+i}$ multiplied by $-a_i s$ and the determinant of the matrix found by crossing out the lowermost row and the i-th column, which turns out to be $(-s)^{\mathrm{N}-1}$. Each of these terms gives a contribution of $-a_i s^{\mathrm{N}-i+1}$, except from the last column, where we have just $(1 - a_\mathrm{N} s)$. All together, we have

$$\begin{aligned}
\mathcal{Z}[y(t)] &= \frac{s^\mathrm{N}}{1 + a_\mathrm{N} s + a_{\mathrm{N}-1} s^2 + \ldots + a_1 s^\mathrm{N}} \\
&= \frac{s^\mathrm{N}}{g(s)}. \qquad (18)
\end{aligned}$$

g(s) is the so-called *characteristic function* of the shift register. It only depends on the position of the taps.

Requiring a Period in $y(t)$. We are now ready to compare the z-Transform of a particular output sequence (18) to the general expression (14) for a certain periodicity: As we start with zeros in the shift register at time $t = 0$, we will see zeros at the output during $\mathrm{N} - 1$ clock cycles, i.e. we have $r_i = 0$ for $i = 0 \dots \mathrm{N} - 1$. The first term in (14) therefore drops out.
At $t = \mathrm{N}$, the injected one appears at the output, i.e. we have $r_\mathrm{N} = 1$. Note that any periodic sequence *must* start exactly at $t = \mathrm{N}$: it is not possible that there is a further 'preamble' after $t = \mathrm{N}$ because every state has a well defined predecessor state (which we can calculate) for this type of register! We want the following sequence to have a period P and denote the output signals after r_N by α_i, i.e. $r_{\mathrm{N}+i} = \alpha_i$. Equation (14) of such a periodic sequence $y(t)$ take the following form (with $\tau = \mathrm{N}$):

$$\mathcal{Z}[y(t)] \;=\; \frac{s^\mathrm{N}}{1+s^P} \sum_{i=0}^{P-1} \alpha_i s^i = s^\mathrm{N} \frac{h(s)}{1+s^P}. \tag{19}$$

$h(s)$ is a polynominal created from the sequence of 1s and 0s with period P. Comparing (18) and (19), which both describe the same output sequence, gives us the condition which must be fulfilled in order for the sequence to have a period P:

$$\boxed{h(s)\ g(s) = 1 + s^P. \tag{20}}$$

As a reminder, $g(s)$ is the characteristic polynominal constructed as defined in (18) from the topology of the LFSR. $h(s)$ is a polynominal reflecting the output sequence. All calculations are on the Galois field GF(2), i.e. modulo 2.

5 Conclusions

From (20) We conclude the following:

1. The shift register can only have a period P if $g(s) = 1 + a_\mathrm{N} s + a_{\mathrm{N}-1} s^2 + \ldots + a_1 s^\mathrm{N}$ divides $1 + s^P$.

2. A register of period P_1, has, of course, also periods $k \cdot P_1$ for all possible k. To make sure that the period P is the smallest period, $g(s)$ may not divide any $1 + s^p$ with $p < P$. When we want to check that, we do not have to verify all possible p, just the prime factors of P.

6 Examples

6.1 N=4, Maximum Length

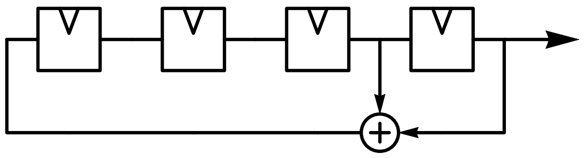

Figure 2: LFSR for N=4 with tap at last position.

The register with N = 4 flipflops in fig. 2 has one 'tap' at the last position, i.e. a characteristic polynominal $g(s) = 1 + s^3 + s^4$.
This $g(s)$ divides $1 + s^{15} = (1 + s)\,(1+s+s^2)\,(1+s+s^4)\,(1+s^3+s^4)\,(1+s+s^2+s^3+s^4)$, but it does not divide $1 + s^5 = (1 + s)\,(1 + s + s^2 + s^3 + s^4)$, the polynominal for a possible smaller period of 5, nor $1 + s^3 = (1 + s)\,(1 + s + s^2)$. The period is therefore 15. Note that the output sequence can be directly obtained from

$$h(s) = \frac{1 + s^{15}}{1 + s^3 + s^4} = 1 + s^3 + s^4 + s^6 + s^8 + s^9 + s^{10} + s^{11}$$

as 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1.

6.2 N=4, Shorter Period

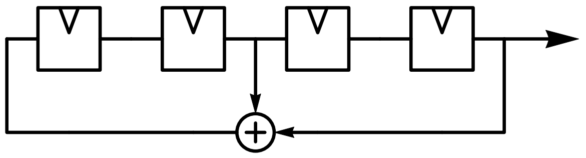

Figure 3: LFSR for N=4 with tap at second last position.

The shift register in fig. 3 uses the second last tap so that we now have $g(s) = 1+s^2+s^4$. From the factorization $g(s) = 1 + s^2 + s^4 = (1 + x + x^2)^2$ we can directly see that this $g(s)$ does *not* divide $1 + s^{15}$ (see the factorization above). If we try to divide $1 + s^k$ by $g(s)$ for smaller $k = 1 \ldots 15$, we find the reminders

1. $1 + x$
2. $1 + x^2$
3. $1 + x^3$
4. x^2
5. $1 + x^3 + x$

6. 0
7. $1 + x$
8. $1 + x^2$
9. $1 + x^3$
10. x^2
11. $1 + x + x^3$
12. 0
13. $1 + x$
14. $1 + x^2$
15. $1 + x^3$

The first division works for a period $k = 6$. Looking at the corresponding polynominal $1 + s^6 = (1 + x)^2 \, (1 + x + x^2)^2$ confirms, that this contains the prime polynominals of $g(s)$.
Note that the next division which works is at $k = 12$, twice the period.

7 N=6, Period is a Fraction of $2^6 - 1$

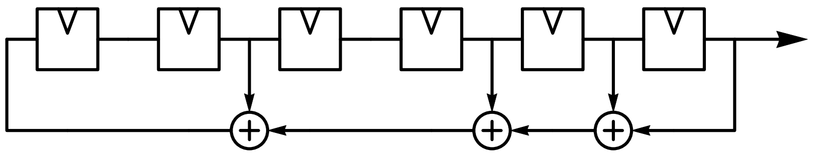

Figure 4: LFSR for N=6.

The longer shift register in fig. 4 has several taps so that $g(s) = 1+s^2+s^4+s^5+s^6$. This characteristic polynominal divides $1 + s^{63}$, so that we have a period of 63. It turns out, however, that this is not the fundamental (lowest) period, by noting that also $1 + s^{21}$ is divided by $g(s)$. There is no lower exponent which works, so that the period of fig. 4 is 21.

8 Fibonacci and Galois Type Registers

So far, we have treated the type of shift registers shown in fig. 1, where the output, the taps and, possibly, an input are all XOR-ed together and fed back to the input. This topology is often called a ‘Fibonacci’ type shift register. In a practical application, it has the drawback that the many daisy chained XOR gates can introduce quite some delay so that the maximum clocking frequency of the register is limited. Another possible topology, the ‘Galois’ type shown in fig. 5 avoids this drawback because there is at most one XOR gate in front of every flipflop.

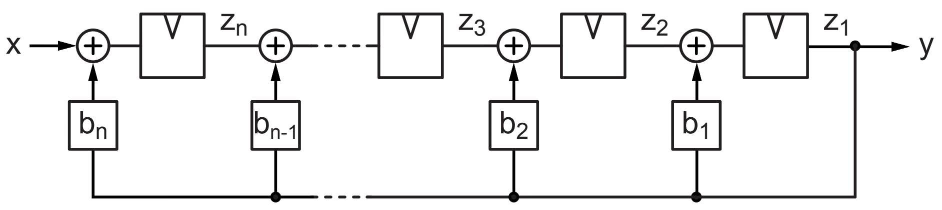

Figure 5: ‘Galois’ type linear shift register

Can we apply what we have learned to this type of register? Let’s write down again the recursion which allows us to calculate $\vec{z}(t+1)$ from $\vec{z}(t)$:

$$\begin{aligned} z_i(t+1) &= z_{i+1}(t) + b_i \cdot z_1(t) \quad \text{for} \quad i = 1 \dots \mathrm{N}-1 \\ z_{\mathrm{N}}(t+1) &= x(t) + b_{\mathrm{N}} \cdot z_1(t) \\ y(t) &= z_1(t). \end{aligned}$$

In Matrix notation, this is

$$\vec{z}(t+1) = \begin{pmatrix} b_1 & 1 & 0 & \cdots & 0 \\ b_2 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ b_{\mathrm{N}-1} & 0 & 0 & \cdots & 1 \\ b_{\mathrm{N}} & 0 & 0 & \cdots & 0 \end{pmatrix} \vec{z}(t) + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} x(t)$$

This looks very similar to (6), just that now, the coefficients with small indices are close to the diagonal. If we do the same as before with this new matrix, it turn out that the characteristic polynomial is the same as before, if we use b_1 for a_{N} and so on! The period of the Galois Register is therefore the same as for the Fibonacci register, if we *reverse the tap order.*

References

[1] M. Gössel, Angewandte Automatentheorie I und II, Wissenschaftliche Taschenbücher Band 116 und 117, Akademie Verlag, Berlin, 1972, ISBN 3528061170

[2] W. S. Golomb, Shift Register Sequences, Aegean Park Press, 1982, ISBN 0894120484

[3] Efficient Shift Registers, LFSR Counters and Long Pseudo-Random Sequence Generators, Xilinx Application Note XAPP 052, July 7,1996 (Version 1.1), http://www.xilinx.com/bvdocs/appnotes/xapp052.pdf