Welcome to the World of CDMA

Linear Feedback Shift Registers

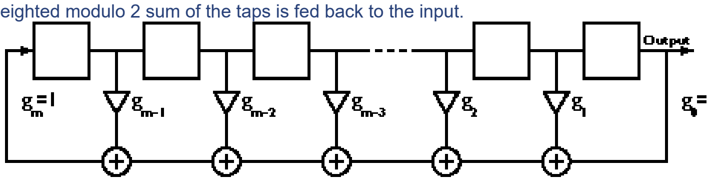
Ideally the spreading codes used in direct sequence spread spectrum systems would be truly random binary sequences, as might be produced by consecutive tosses of an unbiased, memoryless coin. This is not practical. Both transmitter and receiver must generate the same sequence, time-aligned, in order to communicate with one another. Receivers thus must perform synchronization searches by changing their time offset hypothesis until the transmitter timing is located. Achieving high capacity in the CDMA environment also requires that the spreading rate be high: 1.2288 MHz in IS-95A CDMA. If truly random sequences were to be used then they would have to be pre-generated and pre-stored in all transmitters, with a matching copy in all receivers. Deterministic methods of generating the pseudo-random sequences are preferable. The CDMA air interfaces use linear feedback shift register (LFSR) generators for this purpose.

The maximal-length binary sequences produced by linear feedback shift registers are widely used for direct sequence spectrum spreading. There are several reasons for this:

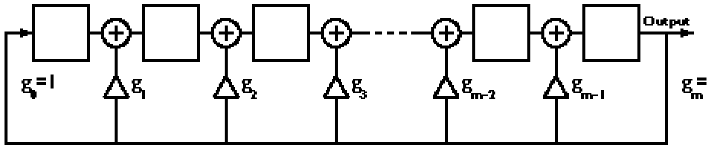
- LFSR sequences are easily generated by very simple binary logic circuits.
- Very high speed generators are possible because of the simple logic.
- Maximal length sequence generators are easily designed using finite (Galois) field mathematics.
- The full period autocorrelation functions of maximal-length LFSR sequences are binary-valued, facilitating synchronization searching.

It is easily shown, by the way, that any linear feedback binary state machine that generates a maximum length output sequence must be equivalent to some maximal length LFSR. See, for example, Peterson, et al.

Linear feedback shift registers can be implemented in two ways (Figure 1). The so-called Fibonacci implementation consists of a simple shift register in which a binary weighted modulo 2 sum of the taps is fed back to the input.



The Galois implementation consists of a shift register, the contents of which are modified at every step by a binary weighted value of the output stage.



If the tap weights are identical, and configured as shown in the figures, then the two implementations will produce exactly the same sequence (this can be verified by simple arguments). Initial conditions required to produce the same phase of the sequence are obviously not identical, however.

There are actually two sequences produced by each of these generators. One is the trivial one, of length one, that occurs in both cases when the initial state of the generator is all zeros. The other, the useful one, has length 2^m-1. Together these two sequences account for all 2^m states of the m-bit state register.

The mathematics of these generators is equivalent to the operation of ordinary algebra applied to abstract polynomials over an indeterminate X, with binary-valued coefficients. This is a finite (Galois) field of order 2^m . Each sequence is based on a *generator* polynomial

$$G(X) = g_m X^m + g_{m-1} X^{m-1} + g_{m-2} X^{m-2} + \ldots + g_2 X^2 + g_1 X + g_0$$

whose coefficients are binary, and are the weights shown in the figures. The polynomial is said to be *primitive* if it does not factor and it divides X^r+1, where r=2^m-1. A primitive polynomial of degree m necessarily has g_m = g_0 = 1. If the generator polynomial in a LFSR is primitive then the sequence produced by that generator has maximum length, which is 2^m-1. Maximal length sequences are sometimes called *m-sequences*. For a discussion of the mathematics of finite fields see, for example, Golomb.

Properties

Some properties of m-sequences:

1. An m-bit register produces a sequence of period 2^m-1.
2. An m-sequence contains exactly 2^{m-1} ones and 2^{m-1} -1 zeros.
3. The sum, modulo 2 of an m-sequence and another phase of the same m-sequence yields a third phase of the sequence.
3a. (A corollary of 3) Each node of the generator of an m-sequence runs through some phase of the sequence.
4. A sliding window of length m, passed along an m-sequence for 2^m-1 positions, will span every possible m-bit number except all zeros once and only once.
5. Define a run of length r to be a sequence of r consecutive identical symbols, bracketed by non-equal symbols. Then in any m-sequence there are:

- 1 run of ones of length m
- 1 run of zeros of length m-1
- 1 run of ones and 1 run of zeros, each of length m-2
- 2 runs of ones and 2 runs of zeros, each of length m-3
- 4 runs of ones and 4 runs of zeros, each of length m-4
- ...
- 2^{m-3} runs of ones and 2^{m-3} runs of zeros, each of length 1

If the sequence is mapped to a binary valued waveform by mapping a binary zero to -1 and binary 1 to +1, then the autocorrelation is unity for zero delay, and -1/N at all other times.

Other curiosities:

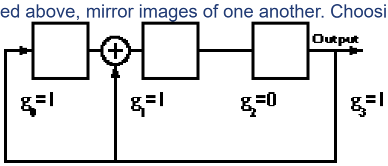
If g(X) is a primitive polynomial of degree m, then $X^m g(X^{-1})=g'(X)$ is also a primitive polynomial. This is easily shown using the definition of a primitive polynomial, and the properties of modulo-2 arithmetic. A little thought shows that the register taps of g'(X) are the mirror image of those of g(X). And the sequence produced by the tap-reversed generator is the time reversal of the original. This seems like it ought to be almost trivially obvious: if time is reversed in a generator, then the arrows in the block diagram are reversed, and in the taps the flow reversal doesn't change the values on any of the three connections!

Example

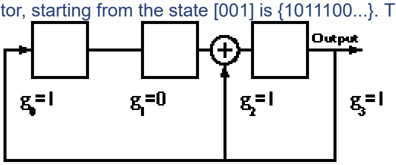
Consider as an example, polynomials of degree 3, so that maximal length sequences are 7 bits long. The primitive polynomials of degree 3 are found as the non-trivial factors of X^r+1, where $r=2^m-1=7$:

$$X^7+1 = (X+1)(X^3+X+1)(X^3+X^2+1)$$

The two degree 3 polynomial factors are, as noted above, mirror images of one another. Choosing the first as g(X), the generator is:



The length 7 sequence produced by this generator, starting from the state [001] is {1011100...}. The other, mirror image, generator



produces {1110100} starting from the state [001].

Offsets and Time Shifts of m-Sequences

The linearity of the m-sequence generators and their properties as a representation of a finite field make it rather simple to offset a state by some prescribed number of states, or to create a transformation matrix that will produce a delayed version of the sequence from an undelayed state register. The Galois generator, in particular, can be regarded as a counter in a Galois field. Counting is equivalent to multiplication of the generator state by a primitive element of the field. The multiplication is equivalent to ordinary matrix multiplication of the state, regarded as a column matrix, by a transformation matrix T.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 & 0 & g_1 \\ 0 & 1 & \dots & 0 & 0 & 0 & g_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & g_{m-3} \\ 0 & 0 & \dots & 0 & 1 & 0 & g_{m-2} \\ 0 & 0 & \dots & 0 & 0 & 1 & g_{m-1} \end{bmatrix}$$

General offsets, say by k states, can be accomplished by calculating T^k and then multiplying the initial state by T^k. In particular, power-of-two powers of T can be pre-calculated, and an arbitrary offset accomplished my multiplying by the powers of T corresponding to the one bits in the binary representation of k.

| Back to Long Code | Back to Short Code |

| Index | Topics | Glossary | Standards | Bibliography | Feedback |

Home • Contact Us • Site Map • Terms & Conditions