An Introduction to Linear Recursive Sequences in Spread Spectrum Systems

By: Richard Schwarz
SIGTEK Inc.

Spread Spectrum technology is being applied to many areas of modern communications such as Cellular Telephones, Global Positioning Satellites (GPS), and Very Small Aperture Satellite Terminals (VSATS) just to name a few. The core concept behind this technology involves spreading the data to be transmitted over a much broader spectrum than what is conventionally done. The spreading is usually achieved by modulating the data with a pseudo random sequence at a much higher frequency. When you demodulate the composite of the spreading code and data with a locally generated, identical, time aligned spreading code the power from the composite bandwidth is collapsed into the data bandwidth. This process yields an effective gain called processing gain. The measure of processing gain is given as the ratio of the data rate to the spreading code rate. Multiple spread users can coexist in the same bandwidth if each user is assigned a different spreading code. Systems utilizing this technique are called Code Division Multiple Access (CDMA) systems. This article focuses on the properties of Linear Recursive Sequence spreading codes and their uses in Spread Spectrum Systems.

Linear Recursive Sequence Fundamentals

A Linear Recursive Sequence (LRS) otherwise known as a Pseudo Noise (PN) sequence is generated from systems which contain three basic elements. These elements are:

1) Delay Elements
2) Linear Combining Elements
3) Feedback Loop Element

The Delay elements are connected in series. Some of the outputs of the Delay elements are combined in Linear Combining Elements with the outputs of other Delay elements and fed back to the input of the first Delay element in the series. This definition can sound intimidating to those of us new to Linear Sequences and Spread Spectrum technology, so let's try to clarify it.

A LRS generator is usually made up of shift registers (the Delay Elements from our definition above) connected in series. Figure 1 shows just such a configuration.

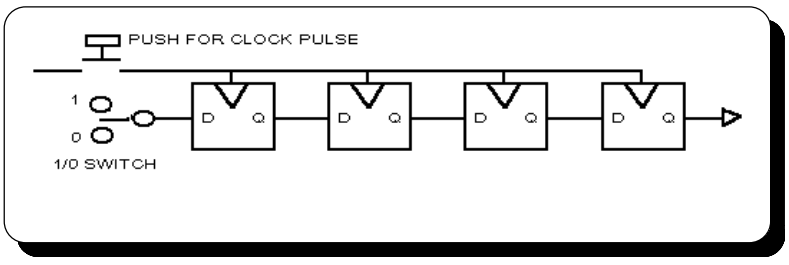

Figure 1 Shift Register Configuration

Each register has an input (labeled D) and output (labeled Q) and a clock input (triangle). Each successive register's input is connected to the previous registers output and all the registers clock inputs are connected to a single clock source so that when a clock pulse is sent the contents of each register is shifted into the next register. As you can see in Figure 1 there is a switch on the input of the first shift register which allows us to set a 1 (high level) or 0 (low level) and a push button switch which is used as our clock. In this example we switch the 1/0 switch to the desired setting for the first register and then press the clock button which sends a clock to all registers and sets the 1 or 0 set on the 1/0 switch into the first register. The process is repeated in our example four times until all four registers are filled with the desired fill. Figure 2 displays the process described above.

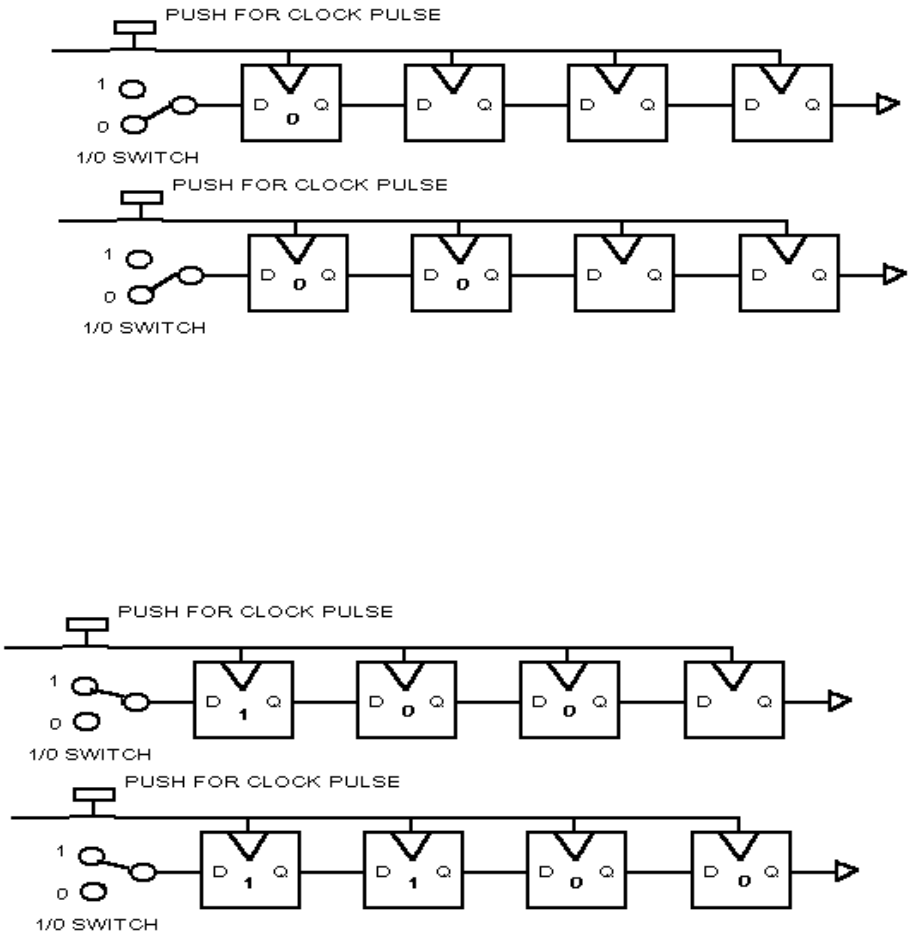

Figure 2 Shift Register behavior during four clock pulses

In our definition of an LRS we talked about three basic elements of which our example above contains only one, the Delay elements. We will now add the *feedback element* which consists of feeding the output of the last register to the input of the first register. Since we have already placed in the desired initial fill contents of the registers we will disconnect the 1/0 switch circuitry. What we now have as seen in Figure 3 is a sequence generator which will generate the initial fill sequence pattern (1100) and will continue to generate this same pattern over and over for as many pushes on the clock button as are made. In this configuration our sequence repeat length is considered to be the same as the number of registers we are using (in our example 4).

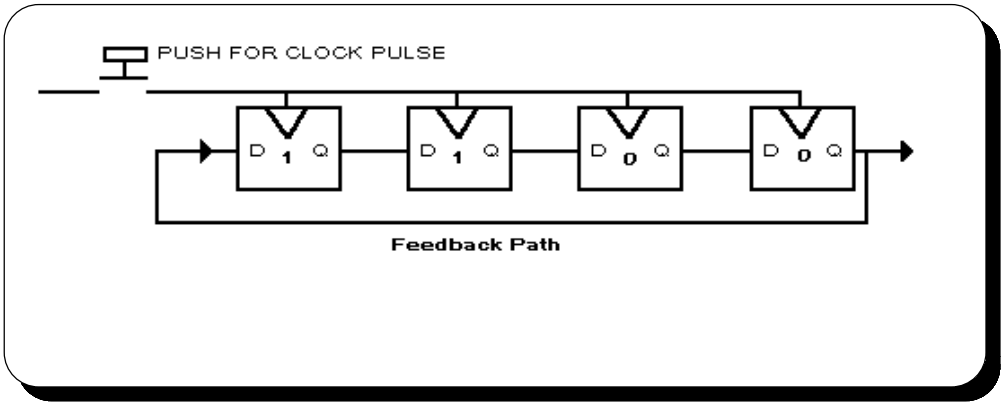

Figure 3 Shift Register with Feedback Path

By adding our third element (the Linear Combiner) we can generate a much longer repeat length using the same number of Delay elements (shift registers). An Exclusive OR (XOR) gate is used as the Linear Combining element whose two inputs are 1) the feedback of the generator and 2) the output of one of the other registers. The output of the Exclusive OR gate becomes the input to the generator. Other TAPS can be added and combined in additional XOR gates if desired but for our example we will only use one. Figure 4 shows this new configuration of our example with the same initial fill we used before. Notice that our sequence repeat length is now 15 as opposed to only 4 from our previous example. As a matter of fact this is the longest sequence that a four register configuration can generate and is equal to 2^n-1 where n = the number of registers used. This configuration can generate sequences with repeat lengths less than $2^4-1=15$ if configured with different TAPS off of outputs of different registers. A configuration which generates a sequence whose repeat length is 2^n-1 and whose register fills contained all possible fill values except the all zero fill after 2^n-1 clocks and in a particular order is called a *Maximal Length* configuration. We will discuss Maximal sequences in more detail later.

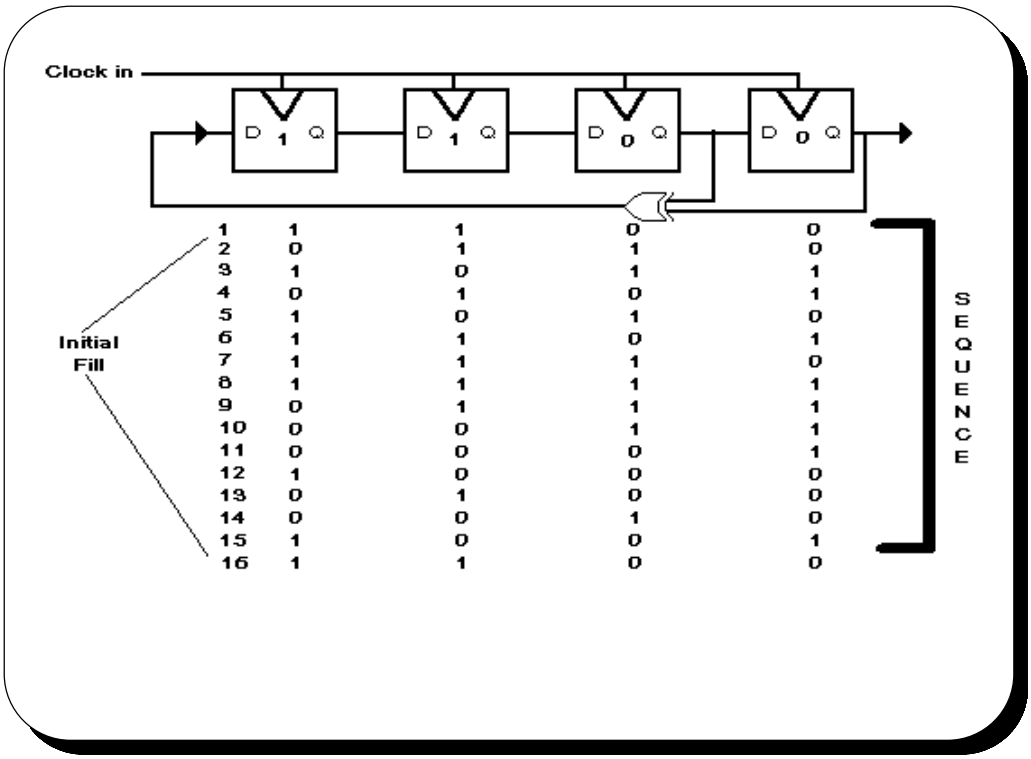

Figure 4 R4 Maximal Length LRS Generator

In our previous example we used a configuration called Out-of-Line Linear Combination. This is where the Linear combiner (XOR gate) outputs are in the feedback path. This is the most common configuration and is often called a SIMPLE-TYPE (S-TYPE) configuration. The alternate configuration is called In-Line Linear Combination or MODULAR-TYPE (M-TYPE) generator. In this configuration the Linear combiners (XOR gates) are in the shift register path. An example of an S-TYPE generator and its equivalent M-TYPE generator are displayed in Figure 6. Notice that the Exclusive OR gates are represented with addition symbols inside of circles. This is to represent the fact that Exclusive OR gates are actually Modulo-2 adders. The truth table for the Modulo-2 Adder is shown below.

1 + 0 = 1	0 + 0 = 0
0 + 1 = 1	1 + 1 = 0

Figure 5 Modulo-2 Truth Table

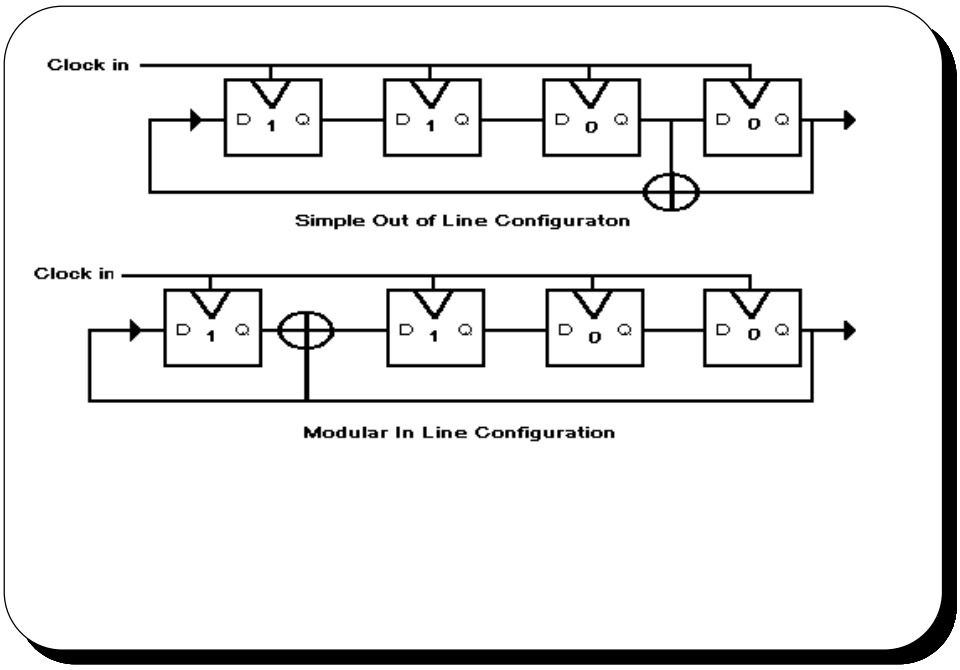

Figure 6 Equivalent In-Line and Out-of-Line R4 LRS Generator Configurations

The LRS Analysis program assumes an S-TYPE (Out-of-Line) configuration. The M-TYPE (In-Line) configuration has some advantages in hardware implementation and is often used to implement the hardware while the S-TYPE is more straight forward for calculation and analysis and is generally more accepted.

Linear Recursive Sequence Generation Parameters

In our examples we will continue to use a four register generator for simplicities sake. It should be noted however that most applications use more registers to achieve longer sequences. Different sequences can be generated by modifying various characteristics of an LRS generator. Three common characteristics which can be modified are the TAP placements, the sequence LENGTH, and the original register FILL contents.

TAPS

As can be seen from Figure 6 LRS Generators have different positions which the non-feedback inputs of the Exclusive OR gates are tapped from. These TAP points are very significant and are the first LRS parameter we will discuss. Let us number our registers from 1 to n where n equals the number of registers (so 1 to 4 in our example) and have register #1's output be our least significant TAP position and the output of our n^{th} register be our most significant TAP position.

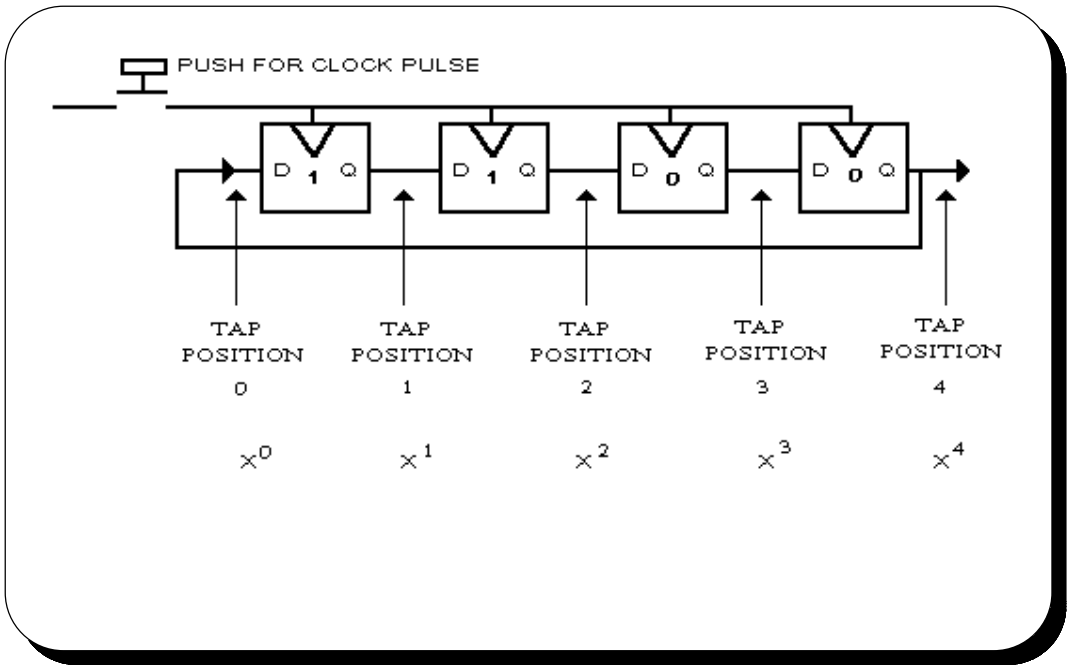

Figure 7 TAP Positions

Further let us represent these positions as a Hexadecimal number. We can then say that we have 2^{n-1} possible tap (ie. $2^3 = 8$) settings since we will always have the Most significant tap which is our feedback path and our least significant TAP which is 0. Figure 8 shows all the possible TAP settings for our example four register (R4) generator. As can be seen from both figures 7 and 8 these TAP settings can also be represented as polynomials where each successive tap (starting from the left) is represented by x raised to its associated TAP position (ie. TAP position 3 = x^3). The zero TAP not shown in the Hexadecimal representations but is shown as $x^0 = 1$ in the polynomial representations.

Figure 8 R4 Tap Settings

There are several aspects of Figure 1.8 that are worth examination. First of all notice that the C h and 9 h TAP settings are equal to 2-1 length and are Maximal length codes. Also notice that

except for the Most Significant Tap (the feedback tap) that 9 h(1001) and C h (0011) are mirror images of each other. That's because for every Maximal length TAP setting there is another Maximal Tap setting that is it's mirror image. Also notice the 0 repeat length TAP settings and notice that each of them has an odd number of TAPs. An XOR gate with an odd number of inputs and the all ones fill would produce a one as the feedback input and never recover from the ones fill. For the same reason an all zeros fill will always produce a zero feedback input and never recover from being all zeros. The zero fill problem applies to odd and even number of TAPs and so is a universal rule. So a rule for our LRS Generator is that all TAP settings should contain an even number of TAPs. There are several other TAP settings in Figure 8 which have a repeat length less than 2^n-1. Maximal length codes have other properties besides their length which make them highly desirable which we will examine later.

LENGTH

The length parameter of an LRS Generator is not as straight forward as one might think. The lengths mentioned in Figure 8 are the natural lengths of the LRS Generator given its particular TAP setting. This is the length which the shift register will run before naturally loading its Initial Fill. Sometimes LRS generators are truncated before reaching their Maximal Length, or are allowed to run several Chips (clock cycles) past its natural length (appended) and are then reloaded with the Initial Fill. Some possible architectures which allow appending or truncating of the Natural length are described below. In Figure 9 a parallel set of registers hold the initial fill and automatically loads this fill in when a clock counter reaches the length set in the counter.

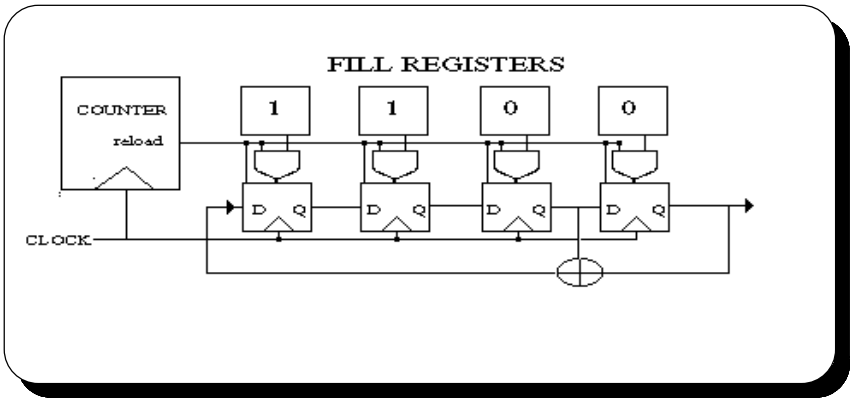

Figure 9 LRS Generator with Length Counter

In another possible architecture the sequence is stored in a RAM chip whose starting address is reset after a predetermined length. Details must be known about the particular LRS generator's architecture before assuming what the length setting means.

FILL

The FILL parameter is the simplest of the three parameters and describes the initial contents of all n registers of an Rn (ie. R4) LRS generator (or the register contents at code phase offset 0). In our example we used a fill of 0011. It is more common to set an impulse (one in least significant

bit of FILL and 0 in all the rest) or an all ones FILL. The all ones FILL is particularly useful as it is easy to spot in the sequence when observing it on an oscilloscope, printout or screen. The all zeros FILL is not valid as it will never recover from this FILL due to the use of XOR gates which use the MODULA-2 addition rule which says that 0 + 0 = 0.

Maximal Length Codes

As mentioned earlier Maximal Length codes are codes whose natural length runs to 2^n-1 elements, where n equals the number of registers used to generate the sequence. Maximal Length codes have four basic properties which are listed below.

1) Two Level Autocorrelation - The autocorrelation of a maximal length sequence consists of two levels. Level one which occurs every 2^n-1 elements has a correlation of 2^n-1. In other words all the patterns match. More will be said about correlation later but for now it is important to realize that if I had two identical Maximal sequences and I slid one sequence by the other sequence one pattern at a time I would get a 100% correlation (match) in the number of chips every time the sequences are exactly time aligned, and about half the chips will match otherwise. This makes Maximal Sequences easier to phase lock than non-Maximal sequences which exhibit partial correlation levels when the sequences being correlated and are not time aligned

2) All possible n-bit numbers-A maximal length code will generate all possible n bit numbers in the coarse of cycling through its frame length. Inside the frame of a maximal length code there are:

1 run of ones of length n.
1 run of zeros of length n-1.
x runs of ones of length n-x.
x runs of zeros of length n-x.

Where n = Length Run and x= 1 to n.

3) Product of identical Maximals with phase offset yields same Maximal with phase offset- When Modula-2 adding (XORing) a maximal length code with a phase offset copy of the same maximal length code you get yet another phase offset copy of the same code as the Product.

4) A Maximal Length code contains one more zero than one in its 2^n-1 elements. This is because an all zero FILL is invalid, and is a FILL which the generator cannot recover from. An all zero portion of the code could be added to RAM generated code however if desired.

The usefulness of these Maximal Codes stem from these four properties mentioned above. In particular the two level Auto-Correlation allows Spread Spectrum receivers to more easily detect code lock of an incoming signal with an LRS code impressed upon it. A basic block diagram of a Direct Sequence Spread Spectrum system is provided in Figure 10.

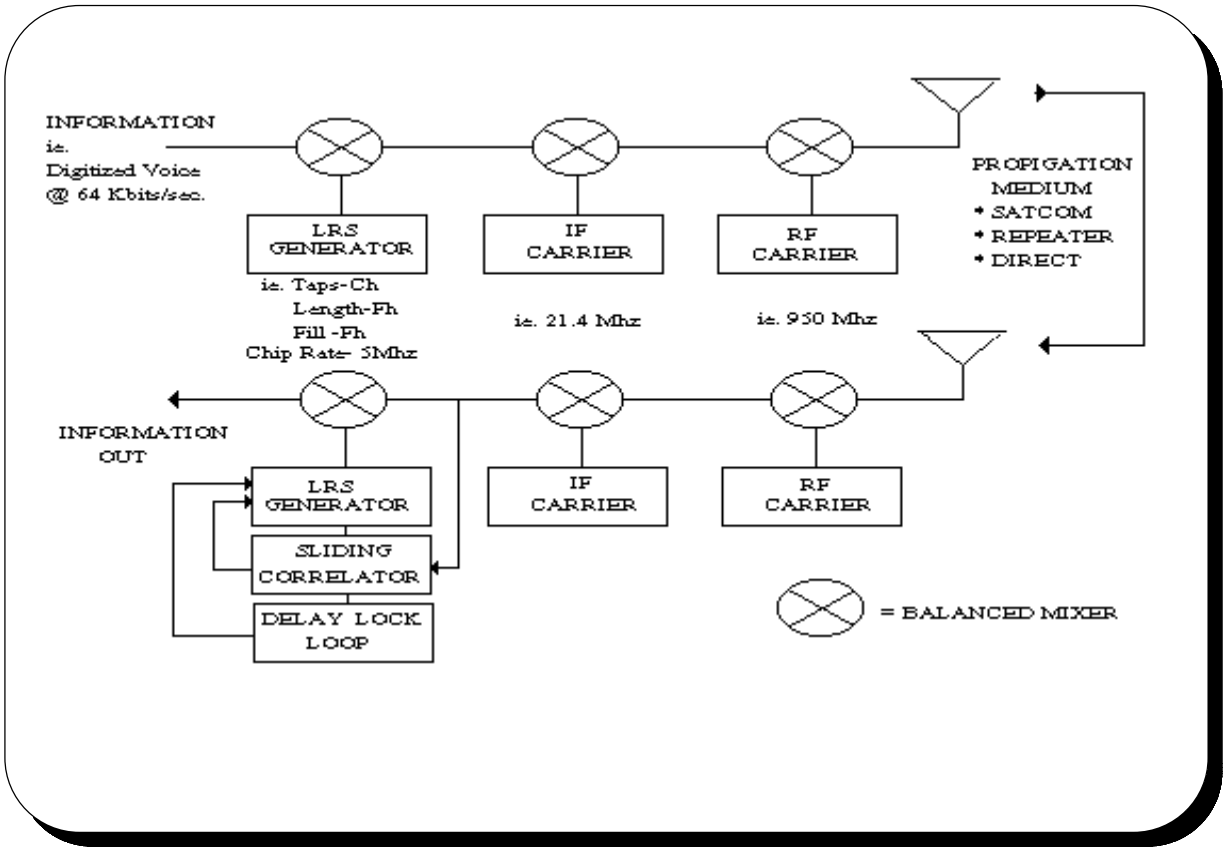

Figure 10 Direct Sequence Spread Spectrum System Block Diagram

In this example narrow band information is mixed with a wideband LRS at the transmitter and then upconverted to an Intermediate Frequency (IF) and then a Radio Frequency (RF) . At the receiver the RF carrier is down converted and then the IF carrier is stripped off. Then the LRS must be removed in order to extract the transmitted information. This is not a trivial task. First the LRS of the incoming signal must be aligned with a locally generated LRS with identical settings (TAPS, LENGTH, FILL, Clock Rate, etc.). Then the local LRS must be phase aligned with the incoming signal. Here's where Maximal codes come in handy. The receiver can slide the local code by the incoming code (usually by dropping clock cycles) and watch the number of bits in the sequence which match until the two sequences match and a strong correlation (match) peak occurs, at which point the receiver stops sliding the local LRS and presumably it is time aligned with the incoming LRS. Then the locally generated code is mixed with the composite incoming signal which strips off the LRS from the incoming signal to produce the transmitted information. Non-Maximal codes produce partial correlations which make it more difficult for the receiver to detect an accurate code phase lock.

Product Codes

A Product Code as discussed in the context of this text is the Exclusive OR-ing (Modulo-2 adding) of two Linear Recursive Sequences to achieve a third sequence which we call the Product Code. The two codes which make up the Product Code are termed as Factor Codes. Product Codes can be made up of more than two Factor Codes but for our discussion we will limit ourselves to two. Figure 11 is a block diagram of a Product generator.

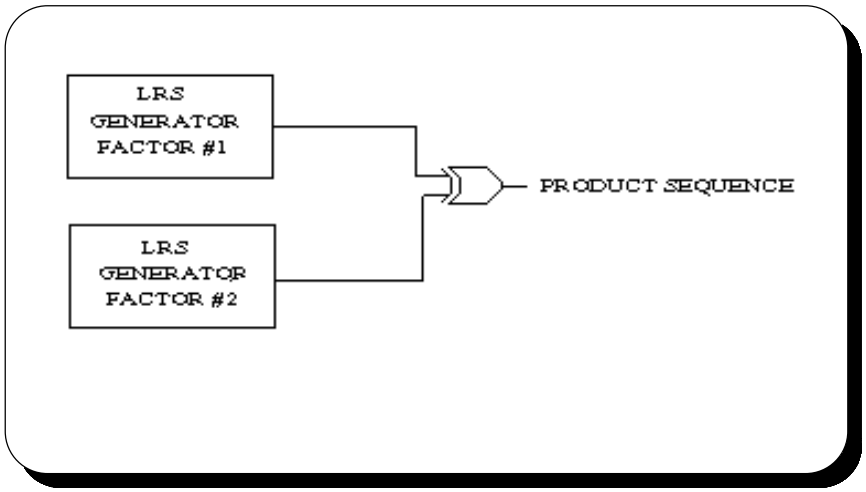

Figure 11 Product Code Diagram

The Natural length of a Product Generator occurs when the Factor Codes sequence's initial fill frame naturally coincide. For example an R4 code whose natural length is 15 and an R5 code whose natural length is 31 will run to a length of 15 x 31 = 465. Another example is a code whose natural length is 15 being XOR-ed with a code whose length is 15 which yields a code with a length of 15.

Factor Codes can be truncated or appended. For example an R4 can be allowed to run past its natural length of 15 to 20 for example and be XOR-ed with an R5 which runs to its natural length of 31. This will produce a Product sequence whose natural length will run to 31 x 20 = 620. If we then truncate the second sequence length (the R5) by one to 30 our Product only runs to a Natural length of 60 because the appended R4 code length of 20 and the truncated R5 length of 30 both have their respective initial fills loaded coincidentally after 60 chips. It must be noted here that Product Codes do not necessarily yield Maximal sequences. They may contain Maximal sequences within them which can be used if the Product Code itself is truncated to the length of the Maximal code portion of the Product Code. Or they may contain a portion of a Maximal sequence or they may themselves be a Maximal sequence. Rule three from our Maximal

Sequence section tells us that the Product of two identical Maximal Sequences with different code offsets (different FILLs) gives us the same Maximal sequence at a different code offset.

Gold Codes

Gold Codes are Product Codes of two different Factor Maximal length sequences with the same lengths. The two Factor Codes are able to generate a family of many non-Maximal Product Codes. An important subset of a family of Gold Codes are Preferred Pair Gold Codes. These are Gold Codes whose cross-correlation spectrum is three valued. These special pairs of Factor Codes with very predictable cross-correlation properties are necessary in an environment where one code must be picked from several codes which exist in the spectrum.. One method used to determine if the code is a Preferred Pair is as follows:

1) Take a length N Maximal Sequence.

2) Create a second sequence by sampling every s^{th} symbol of the first sequence. So now sequence #2 is a s^{th} decimation of sequence number 1. This second code must also be a Maximal code of the same length.

3) The length should be odd and the gcd(N,s) = 1. (they must be prime).

The relationship of Product Codes / Gold Codes / and Preferred Pair Gold Codes is shown in figure 12. For example codes which are mirror images of one another can be used to generate a Gold Code but cannot be used to generate a Preferred Pair Gold Code.

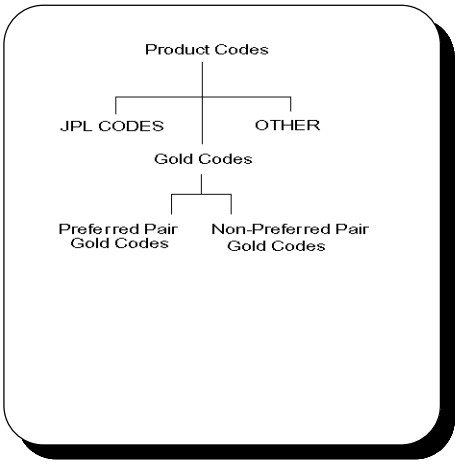

Figure 12 Product Code Relational Diagram

The number of three value spectrum cross-correlated Product codes that can be generated with a pair of Maximals is equal to the number of phase differences which can be set between the two generators by adjusting the fills of one or both generators. For a Gold code generator using two R7 Factor generators the number of different combinations equals $(2^7-1) = 127$ Non-Maximal Product code possibilities plus the two original Maximal codes for a total of 129 codes. For a

CDMA system this means 129 different code possibilities 127 of which have a cross-correlation spectrum which is three valued. Correlation (both Auto and Cross) is covered next .

Correlation

Correlation of two sequences can be described as the comparison of the two sequences to see how much they correspond with one another. Various parameters effect the correlation of two sequences including the code lengths, code phases, and the Chip Rate of each sequence. We will demonstrate by correlating a simple square wave with itself. The act of correlating a signal through all phase variations of itself is called autocorrelation For our square wave we begin by separating our wave into two waves each identical to one another. One of the wave forms will remain constant while the second wave form will be slid past the first wave form and a comparison made at each step to determine the amount of correspondence between the two wave forms (# of Agreements - # of Disagreements). The output wave form reflects the amount of correspondence as a rising or declining peak Figure 13 shows our square wave being correlated in quarter cycle steps. The correlation wave form is at its greatest peak when the two wave forms are time aligned, and is at its lowest when the two are 180 degrees out of phase as seen in Figure 14.

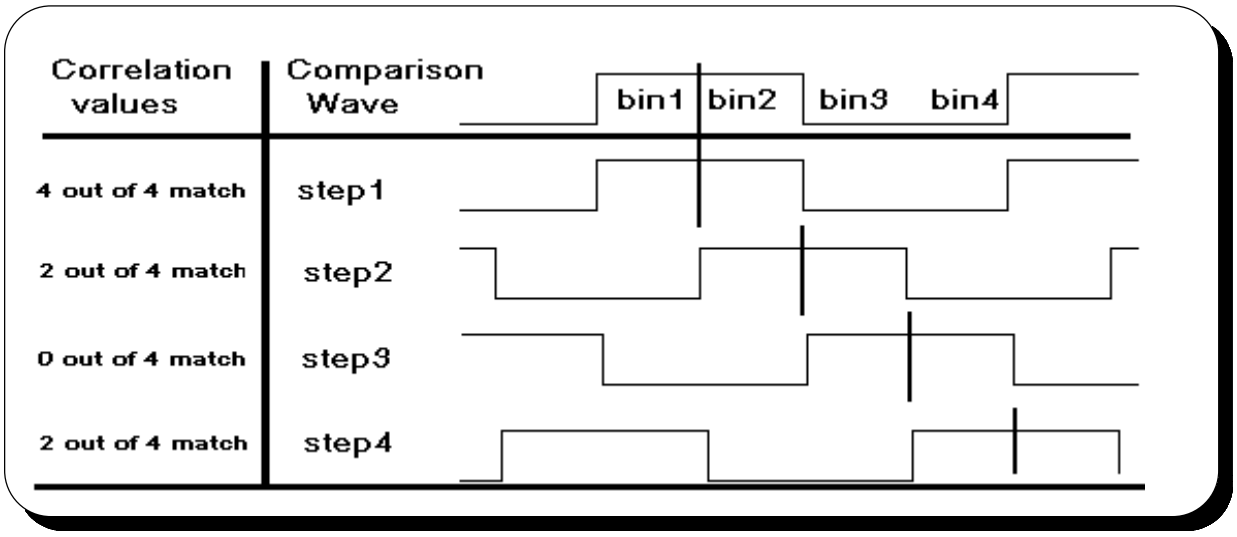

Figure 13 Step Autocorrelated Square Wave

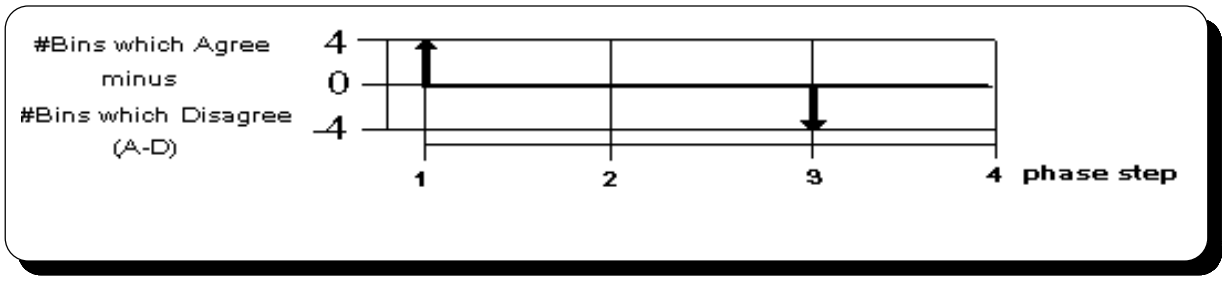

Figure 14 Step Correlation display of Autocorrelated Square Wave

In the previous example we used a discrete step size of one quarter cycle in the phase sliding of our correlation. A digital correlator will always have some step size associated with it. An analog correlator (an integrator) however can correlate over the entire range of phase offsets. The autocorrelation function is in fact an integration of a function (signal) from one period in time (phase) to another, and is given by:

$$R_f(t) = \lim_{\to\infty} \int^{} {}^{*}( )\,(+\tau)$$

Doing our same example in this format and limiting our operation to only one clock cycle (--<τ<) we have:

$$R_f(t) = \int^{-+\tau} = (-+\tfrac{\tau}{})$$

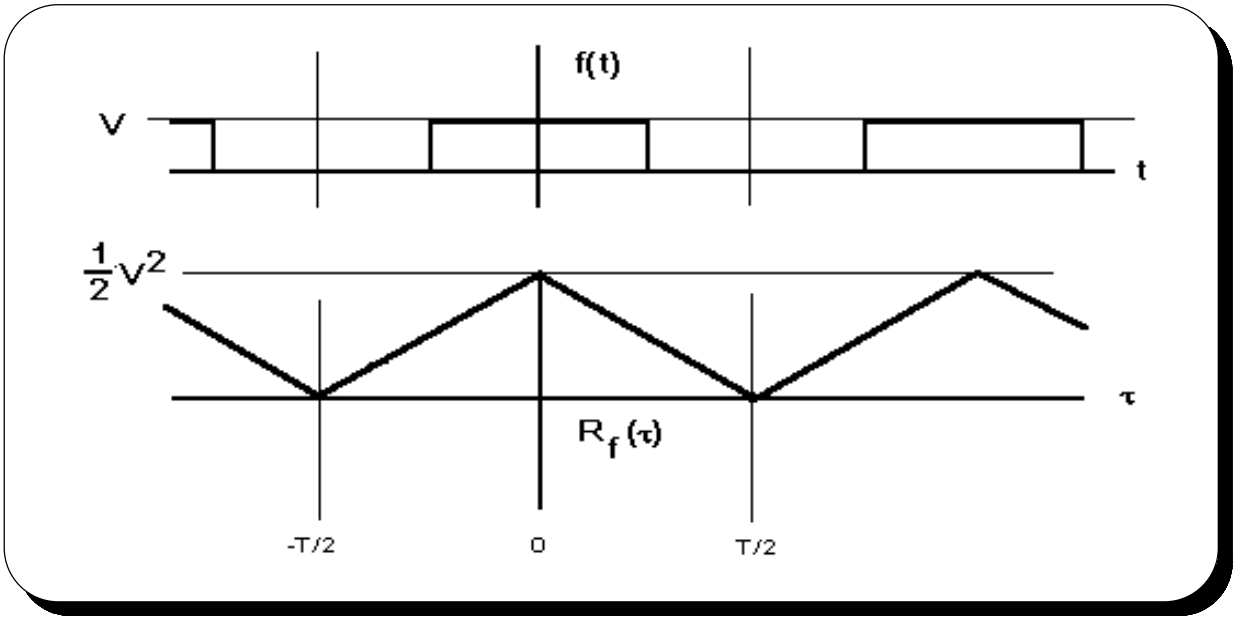

Figure 15 Square Wave and its Autocorrelation function

From Figure 15 it can be seen that the Correlation peak is at its maximum (.5 V²) at t = 0. As the replica wave form is shifted in time the waves no longer correspond in time and the correlation peak starts to drop until the replica wave is 180 degrees out of phase with the original wave (t = T/2). At this point our correlation is 0. Because the wave form is repetitive, the wave form starts to move toward realignment right away and the correlation peak rises. Depending on the step size of the correlation process the correlation peak can look like anything from a triangle to a line peak.

Cross-correlation

In cross-correlation the two compared wave forms are different wave forms. This gives us an idea of how similar the wave forms are to one another at different phases. Correlation (both Auto and Cross) involving two sequences is done as follows: The first wave form will be held at a constant phase while second wave form is slid by it in one chip steps. At each phase step a count is taken of how many chips match (agree) and how many chips do not match (disagree). Then the number of disagrees is subtracted from the number of Agrees (ie. A-D) to yield the correlation value for the two sequences at this particular phase offset.

If we cross-correlate the R4(TAPS = Ch) sequence from chapter one with both itself (autocorrelation) and then with another R4 (TAPS = 9h) sequence (cross-correlation) we would see that the autocorrelated R4s have only two values; 15 and -1. Every time the two sequences are perfectly time aligned they have all fifteen patterns matching exactly. In this case the Agrees=15 and the Disagrees =0 so A-D = 15. At all other phases the correlation value is 7-8= -1. In the cross-correlation example we get correlation values ranging from 7 to -5. These correlation values can be written as power ratios using the following formula:

$$\text{Correlation in db} = \frac{|\quad (\ -\ )|}{(\ +\ )}$$

The power ratio correlation shown here is incoherent. That is to say that it is irrelevant whether the sequences are inverted or not. A correlation value of -15 is at the same power level as +15 since the numerator of the equation is in Absolute Value brackets. The LRS Analysis Correlation Screen is displayed in a logarithmic format with the polarity of the correlation (positive or negative) shown only in the Starting Phase # Correlation Value display. A logarithmic value representing the power ratio correlation of the first displayed phase is also shown which is calculated from the formula shown above and will be the same for two values with equal magnitudes and different polarities.

Partial Correlations

Partial Correlations are correlation levels which occur between when the Maximum number of chips which match in a frame, and when the zero correlation baseline level occurs. For Maximal sequences this zero correlation baseline is the number of chips which should correlate when the sequences are not time aligned, and is about half +- 1 chips in the sequence frame.

Partial Correlations can occur when Non-Maximal sequences are used for Spreading Codes, or when other sequences are present in a CDMA (Code Division Multiple Access) environment, and cross-correlate to some extent with a receivers locally generated sequence. Sometimes, in order to speed up the correlation process only a portion of the sequence is checked to determine receiver lock. This does indeed speed up the process but can also allow partial correlations of the portion of the code which is used. Predictable Partial Correlations can be used to your advantage.

In particular extremely long sequences with partial correlations with predictable levels and phases can aid in quick code lock times. This technique is used in JPL Ranging codes.

References:

Rodger E.Ziemer, Rodger L. Peterson, Digital Communications and Spread Spectrum Systems, (New York, Macmillan Publishing Company, 1985) .

Robert C. Dixon, Spread Spectrum Systems, (John Wiley & Sons, Inc, 1984).

W.W. Peterson, E.J. Weldon, Jr. , Error Correcting Codes (Massachusetts Institute of Technology, 1972).

Ferrel G. Stremler, Communications Systems, (Addison-Wesley Publishing Company, 1990).

Richard Schwarz is a Senior Engineer at SIGTEK Inc. He can be reached at 410-290-3918 or at:

rdschwarz@SIGTEK.COM