Chapter 1

Spread Spectrum Coding

1.1 Introduction

This chapter introduces the concepts behind pseudo-random noise (PN) generation, and discusses the requirements of such a sequence generator in terms of a spread spectrum communications system. Only codes useful for communication systems will be discussed here, although PN sequences are commonly used in a variety of situations such as ranging and error checking. The codes used in spread spectrum systems are inherently much longer than those found in other systems as they are intended for bandwidth spreading rather than information transfer.

A spread spectrum system is largely categorised by it's coding scheme. The type of code employed, it's length, and it's chip-rate all define the overall system parameters. In order to alter the system's spreading capability it is necessary to alter the coding arrangement.

Maximal length sequences will be discussed, along with a method which allows maximal codes to be combined to create Gold codes. Auto and cross correlation will be introduced, and mention will also be made of other, not so common, spreading codes such as Kasami sequences. Error correcting codes and methods of implementing them will be discussed, with details given on block codes, convolutional coding, the Viterbi algorithm and Reed-Solomon coding.

1.2 Maximal Length Sequences

A maximal code is defined to be the longest code that can be generated by a given shift register of a given length. For a binary shift register sequence generator, the maximum sequence length is given by $2^n - 1$, where n is the number of stages in the shift register. A shift register sequence generator consists of a logical combination

of a select number of output stages being fed back to the input of the shift register. An example is shown in Figure 1.1, where arbitrary feedback has been shown.

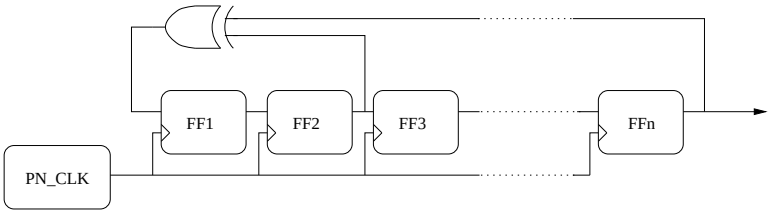

Figure 1.1: Generic maximal length shift register sequence generator

Maximal code sequences possess a number of important properties which are useful in spread spectrum applications. Each of these properties will be dealt with in more detail in the following sections.

- Carrier Balance
- Linear Addition
- State Exhaustion
- Auto and Cross Correlation

1.2.1 Carrier Balance

Maximally generated codes are balanced. That is, the number of ones in a sequence exceeds the number of zeros by one. This is essential since it ensures that any DC component of the sequence can be neglected, as any code imbalance will be clearly obvious as unwanted peaks in the resulting spectrum; thus giving a clear indication of the code clock rate. Such a feature is undesirable as it allows a third party to locate, and possibly interfere with, the transmitted signal. Since one of the motivations behind a spread spectrum system is to conceal the signal, there is little point in advertising it's presence with clearly visible peaks at multiples of the chipping rate. Using a bi-phase encoding scheme of $\pm M$, a code's DC offset is defined to be inversely proportional to it's length, or $\frac{M}{2^n-1}$. Thus, the longer the code sequence, the lower the effect of that sequence on carrier balance.

Figure 1.2 shows the output spectra of a typical direct sequence, spread spectrum system. In this case, a 9600 baud digital signal was spread using a 4 stage maximal length shift register. This PN code generator had a code length of $2^4 - 1 = 15$, and was clocked at 1MHz, thus giving a process gain of $\frac{10^6}{9600} = 104.2$. The PN modulated signal was then mixed with a 10MHz RF local oscillator. A block diagram for this system is shown in Figure 1.3.

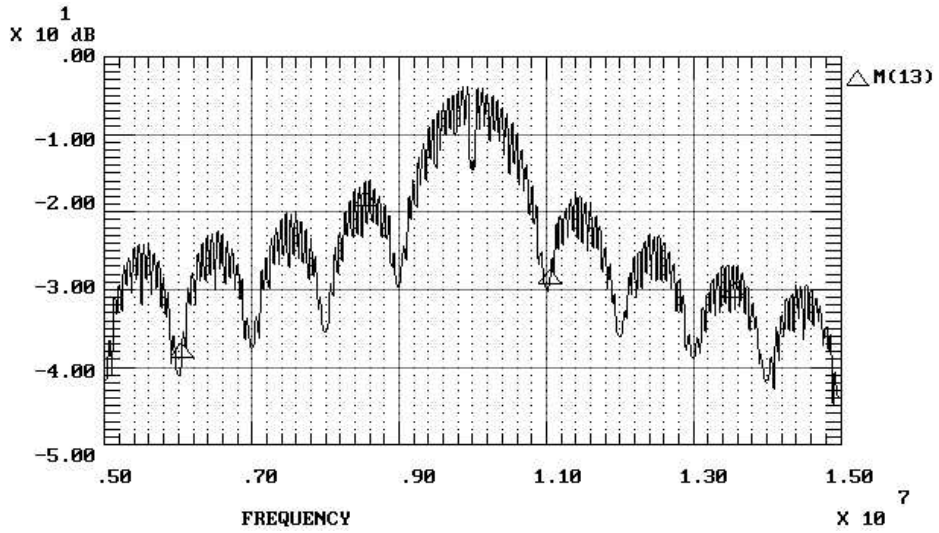

Figure 1.2: 9600 baud data spread using a 1MHz PN clock

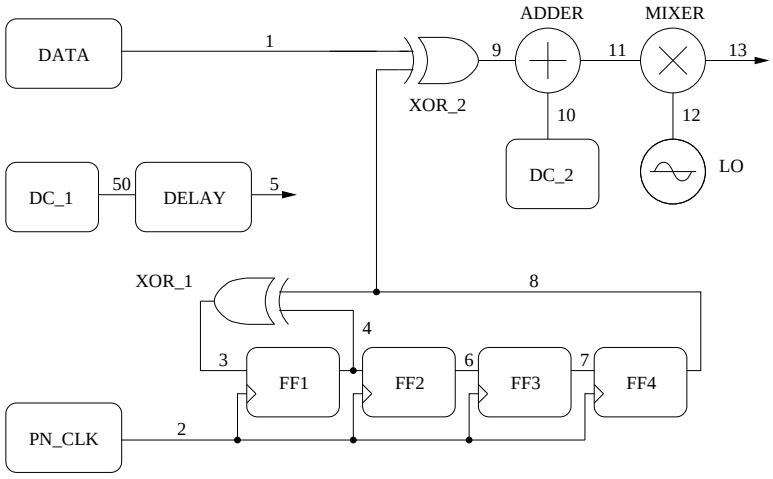

Figure 1.3: Block diagram of 9600 baud DS SS transmitter

Note that the bandwidth of the main lobe centered around 10MHz is exactly twice the PN code clock rate. In this case, the main lobe has been spread from 9MHz to 11MHz, with 93% of the transmitted energy being contained within this bandwidth. It is usual to specify the transmission bandwidth as the first null-to-null bandwidth, with all extraneous information contained outside this region being ignored [8].

Figure 1.4 is a more detailed view of the main lobe bandwidth where minor peaks and nulls are visible as part of the main spectrum. These nulls are spaced at regular intervals, and are given by the relation

$$\text{Null Spacing} = \frac{\text{PN Clock rate}}{\text{Number of bits in the PN sequence}} \tag{1.1}$$

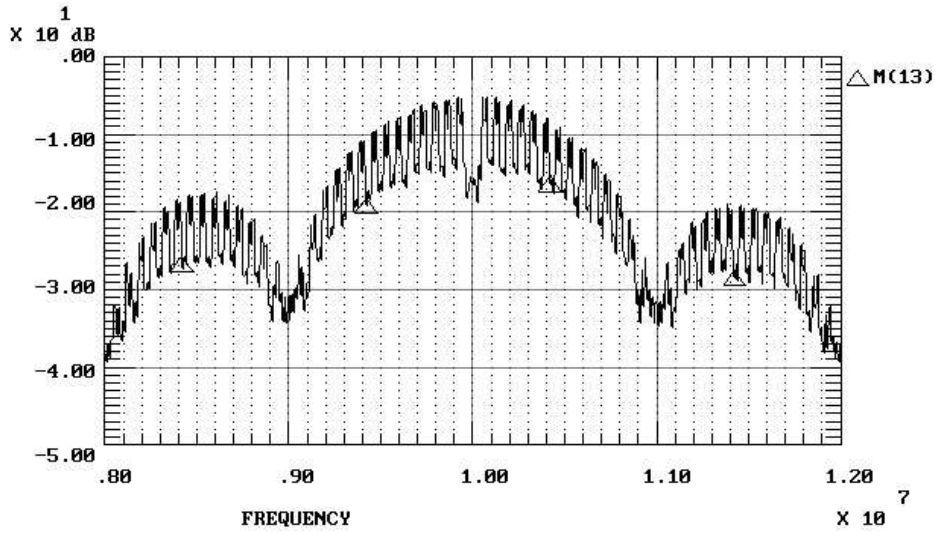

Figure 1.4: Main lobe showing suppressed carrier

In this example the nulls occur every $\frac{10^6}{15} = 66.6$kHz. Also, note the large null at 10MHz. This suppressed carrier is a desirable effect and is due to the balanced nature of the PN code - the number of ones and zeros in the sequence being equal. If the TTL to bipolar stage in Figure 1.3 is removed, an unbalanced condition results; with a prominent carrier present. This effect, shown in Figure 1.5, is highly undesirable, as previously explained. Even though the PN code has not been changed, and the spreading code itself is still balanced, a DC offset exists because of the non-balanced input data. Another side effect of the non-balanced condition is the obvious rise in signal level. With a balanced spreading code, the main lobe amplitude is approximately 8dB lower than that for a non-balanced signal. As measured from the first null position, the amplitude of the balanced code ranges from -34dB to -5dB, a relative difference of 29dB. When a non-balanced code is used, there is still 29dB of relative difference, but the magnitude varies from -26dB to +3dB. This undesirable rise in signal level further shows the benefits of using a balanced PN code.

1.2.2 Linear Addition of Sequences

Adding a maximal length linear code using modulo-2 addition with a phase offset replica of itself gives another replica with a phase offset which differs from either of the two original sequences. This property allows a designer to choose any desired code phase or offset from the original sequence. A typical application of this is in operating multiple correlators to reduce the effective time required for synchronisation. A common system currently using the same code for modulation by all transmitters

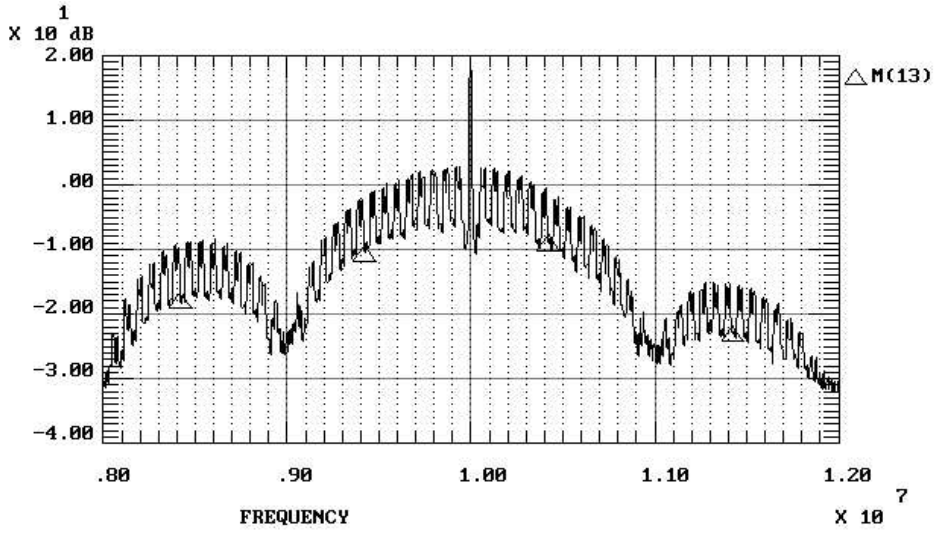

Figure 1.5: Result of non-balanced spreading code

is the global positioning system or GPS. In the GPS system, each transmitter uses the same code, but with a phase offset, thereby preventing the receiver from synchronising to more than one transmitted code at any one time. The autocorrelation property of the sequence is used here to ensure good effective signal orthogonality.

The addition of two $m-$sequences, each of length r, produces a sequence also of length r, but it is not maximal. This is one of the most valuable properties of linear addition. The resultant sequence itself however, is different for each phase offset combination of the original $m-$sequences. Thus, a pair of linear sequence generators, each of length r, can generate r non-maximal linear codes, each of length r. If the original $m-$sequences are chosen properly, then the resultant set of r composite sequences will have low and equal autocorrelation values. For example, a pair of n stage shift register generators, as shown in Figure 1.6, would be capable of producing $2^n - 1$ different non-maximal linear codes, each of length $2^n - 1$, in addition to the two basic linear maximal codes.

1.2.3 State Exhaustion

Every possible state of a given length sequence generator will exist at some time during a single cycle. Each state will only exist for one clock period with the exception of the all-zero state which must not be allowed to occur. When every flip-flop in the register is set to zero, a locked state occurs, since the modulo-2 addition of any combination of zero always results in zero. A binary shift register generator with maximally connected feedback cycles through $2^n - 1$ states. These states are known as $n-$tuples, and an example of a three stage register is shown in

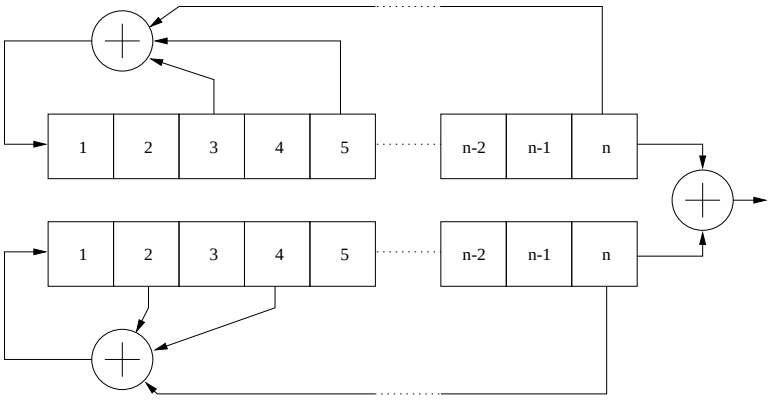

Figure 1.6: Shift register generator for $2^n - 1$ non-maximal codes

Table 1.1. The pseudo-random manner in which the $n-$tuples are ordered is typical of all maximal length sequences, and is dependent on the feedback used.

Table 1.1: Binary to decimal conversion of code $n-$tuples (three stage register) showing pseudo-random ordering

State	3-Tuple	Decimal Equivalent
1	1 1 1	7
2	0 1 1	3
3	0 0 1	1
4	1 0 0	4
5	0 1 0	2
6	1 0 1	5
7	1 1 0	6
8	1 1 1	7

1.3 Auto and Cross Correlation

The general definition of autocorrelation is given by

$$\psi(\tau) = \int_{-\infty}^{+\infty} f(t)f(t-r)dt \tag{1.2}$$

and is an indication of the similarity between a signal and a phase-shifted replica of itself. Calculating this value is important when deciding upon codes which give the least probability of false synchronisation.

The autocorrelation of a maximal length linear sequence is such that the correlation value is -1 for all values of phase shift except for the 0 ± 1 chip phase region, where the autocorrelation follows a linear relationship from -1 to the peak amplitude

of $2^n - 1$ (the code length). A 255-chip maximal length sequence $(2^8 - 1)$, therefore, has a minimum to peak autocorrelation range of 256, or $10\log_{10}(256) = 24.1$ dB. Such a peak is an invaluable property as it allows a receiver to discriminate between wanted and unwanted signals using a clearly defined criteria.

Cross correlation is of interest in areas such as Code Division Multiple Access (CDMA) networking. In a CDMA network, the receiver must not respond to a signal which does not conform to the desired sequence, so a low cross correlation value between each of the network codes is required. The cross-correlation is a measure of the degree of similarity between two different codes. When a large number of transmitters using different codes are to share a common frequency band, as in a CDMA network, the code sequences must be chosen with a great deal of care in order to avoid interference between users. A high degree of correlation between a receiver's reference signal and an unwanted code results in an increase in the number of receiver false alarms and could possibly lead to false synchronisation. This is undesirable.

The bandwidth spreading waveform, which is controlled by a pseudo noise sequence, is given by

$$p(t) = \sum_{i=-\infty}^{\infty} p_i \psi(t - iT_c) \tag{1.3}$$

where $p_i = +1$ if the particular symbol, a_i, of the pseudo noise sequence is a 1, and $p_i = -1$ if it's a 0. The chip waveform pulse $\psi(t)$, is usually confined to the interval $[0,T_c]$, and this is one of the assumptions we make when deriving the autocorrelation of $p(t)$ [33]. We shall also assume the sequence is maximal, the periodic spreading waveform is infinite, and the chip waveform definition is such that

$$\psi(t) = \begin{cases} 1, & 0 \leq t < T_c \\ 0, & \text{otherwise} \end{cases} \tag{1.4}$$

In general, the autocorrelation of a periodic waveform $x(t)$ with period T is defined as

$$C_x(\tau) = \frac{1}{T} \int_{-T/2}^{T/2} x(t)x(t+\tau)dt \tag{1.5}$$

where τ is the relative delay variable. It follows that $C_x(\tau)$ has period T. If the maximal sequence has length $L = 2^n - 1$, then $p(t)$ has period LT_c, and (1.3) to (1.5) give the autocorrelation of $p(t)$ such that

$$C_p(jT_c) = \frac{1}{L} \sum_{i=1}^{L} p_i p_{i+j} = \frac{A_j - D_j}{L} \tag{1.6}$$

where A_j denotes the number of terms with $p_i p_{i+j} = +1$ and D_j is the number of terms with $p_i p_{i+j} = -1$. The correspondence between p_i and a_i implies that A_j equals the number of 0's in a period of $\{a_i + a_{i+j}\}$. Thus, $A_j = (L-1)/2$ and, similarly, $D_j = (L+1)/2$ if $j \neq 0$, (modulo L). A is an expression of the number of agreements in the autocorrelation, whereas D indicates the number of disagreements. When $j = 0$, (modulo L), $C_p(jT_c) = 1$. Therefore,

$$C_p(jT_c) = \begin{cases} 1, & j = 0 \pmod{L} \\ -\frac{1}{L}, & j \neq 0 \pmod{L} \end{cases} \tag{1.7}$$

A time lag can be expressed in the form $\tau = jT_c + \epsilon$, where j is an integer and $0 \leq \epsilon < T_c$. After substitution of (1.3) and (1.4) into (1.5), the resulting integrals can be evaluated in terms of $C_p(jT_c)$. Using (1.6), we obtain

$$C_p(\tau) = C_p(jT_c + \epsilon) = \left(1 - \frac{\epsilon}{T_c}\right) C_p(jT_c) + \frac{\epsilon}{T_c} C_p(jT_c + T_c) \tag{1.8}$$

By substituting (1.7) into (1.8) we can find $C_p(\tau)$ over one period, which is given by

$$C_p(\tau) = \begin{cases} 1 - \left(\frac{L+1}{L}\frac{|\tau|}{T_c}\right), & |\tau| \leq T_c \\ -\frac{1}{L}, & T_c < |\tau| \leq LT_c/2 \end{cases} \tag{1.9}$$

Since $C_p(\tau)$ has a period LT_c, it is specified completely by this equation. The autocorrelation can be expressed in a compact form as

$$C_p(\tau) = -\frac{1}{L} + \frac{L+1}{L} \sum_{i=-\infty}^{\infty} \Lambda\left(\frac{\tau - iLT_c}{T_c}\right) \tag{1.10}$$

An example auto-correlation plot is shown in Figure 1.7

Calculating the Fourier transform of the triangular function shown in Figure 1.7 gives,

$$F\left\{\Lambda\left(\frac{t}{T}\right)\right\} = \int_{-\infty}^{\infty} \Lambda\left(\frac{t}{T}\right) \exp(-j2\pi ft) dt = T \ \mathrm{sinc}^2 fT \tag{1.11}$$

where $j = \sqrt{-1}$ and $\mathrm{sinc}\ x = (\sin \pi x)/\pi x$. Because the summation in the right hand side of (1.10) is a periodic function, it can be expressed as a complex exponential Fourier series. We then take the Fourier transform of this series, and express the Fourier coefficients as Fourier transforms, and finally make use of (1.11). This results in

$$F\left\{\sum_{i=-\infty}^{\infty} \Lambda\left(\frac{t - iLT_c}{T_c}\right)\right\} = \frac{1}{L} \sum_{i=-\infty}^{\infty} \mathrm{sinc}^2\left(\frac{i}{L}\right) \delta\left(f - \frac{i}{LT_c}\right) \tag{1.12}$$

where $\delta()$ is the Dirac delta function. The preceding results can be used to determine $P_p(f)$, the power spectral density of $p(t)$, which is defined as the Fourier transform of $C_p(\tau)$. Taking the Fourier transform of (1.10), substituting (1.12), noting that the Fourier transform of a constant is a delta function, then rearranging the result, we obtain

$$S_p(f) = \frac{L+1}{L^2} \sum_{i=-\infty, i\neq 0}^{\infty} \operatorname{sinc}^2\left(\frac{i}{L}\right) \delta\left(f - \frac{i}{LT_c}\right) + \frac{1}{L^2}\delta(f) \tag{1.13}$$

This function is shown in Figure 1.8, where the delta functions are separated by $1/LT_c$.

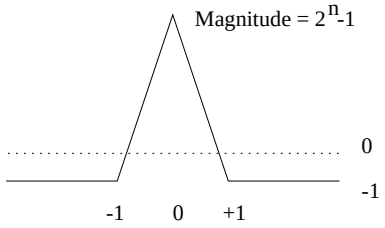

Figure 1.7: Autocorrelation function for a general $m-$sequence

Auto and cross correlation values are expressed as the number of agreements minus the number of disagreements, when two code sequences are compared chip by chip. This is shown in Table 1.2, where the autocorrelation values for all cycles of a three-stage linear $m-$sequence generator are given. This example assumes the reference sequence to be the first one, or stage 0.

Table 1.2: Auto correlation of a 3 stage $m-$sequence generator

Offset	Sequence	Agreements (A)	Disagreements (D)	A-D
0	1110100	7	0	7
1	0111010	3	4	-1
2	0011101	3	4	-1
3	1001110	3	4	-1
4	0100111	3	4	-1
5	1010011	3	4	-1
6	1101001	3	4	-1

It is interesting to note that the net correlation is $2^n - 1 = 7$ for the zero-offset condition, and -1 for all other offsets. This is the typical behavior of $m-$sequences. In the region between the ± 1 offset and zero, the correlation function varies linearly, giving rise to the triangular shape shown in Figure 1.7.

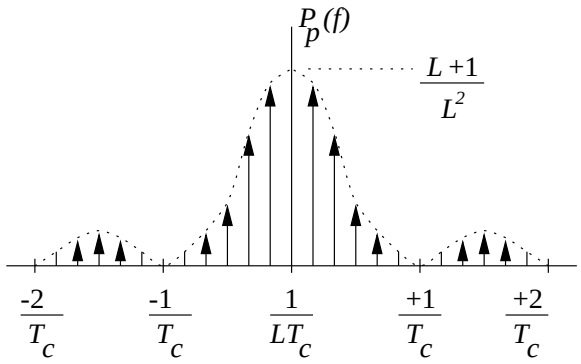

Figure 1.8: Power spectral density plot of a maximal $m-$sequence

The characteristic function shown in Figure 1.7 is only applicable to $m-$sequences though. When other coding schemes are used, the autocorrelation properties may vary markedly. A typical autocorrelation function for a non-maximal sequence is shown in Figure 1.9. The amplitude and position of the minor peaks depend on the code used, and are caused by partial correlation. These minor peaks lower the discrimination margin of the receiver, since it must now decide between the major peak occurring at the zero offset and any of the minor peaks. Non-maximal sequences often have high minor autocorrelation peaks, so for this reason, use of non-maximal sequences for spread spectrum communications systems is not recommended.

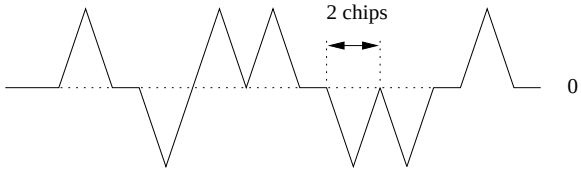

Figure 1.9: Typical autocorrelation function for a non-maximal sequence

Judge [34] considered code division multiplexing by using linear maximal sequences, and states that for two equal-power signals multiplexed together, the signal to noise ratio in each receiver becomes

$$\frac{S}{N} = \frac{S}{(K_1^2 + \frac{T_0}{T})^{1/2}} \tag{1.14}$$

where

T = the cross correlation integration period,

T_0 = the code bit period,

K_1 = value of DC correlation.

This result shows that some Mersenne prime sequences[1] exhibit cross correlation

[1]Those codes for which the code length, $L = 2^n - 1$ is a prime number

values superior to others, sometimes even for non prime sequences longer than prime sequences.

1.4 Gold Code Sequences

Gold codes are ideally suited for use in a CDMA network because of their near optimum cross correlation properties [35]. In addition to this, they also possess the desirable characteristic of being able to generate a large number of different sequences for a given polynomial combination.

It has been established that Gold codes have three–level cross correlation values [36]. If we assume a generator with n stages, producing a PN code of length $L = 2^n - 1$, then it is seen that when n is even (and not 0 mod 4), 75% of the codes generated have a cross correlation bound at $-1/L$.

Gold codes are readily constructed using two related maximal length sequences (also known as $m-$sequences). It is generally accepted that two fixed length shift registers are used, each with fixed feedback taps. The position of these taps is derived from the characteristic phases of the chosen maximal sequences, and determines the overall characteristics of the code generator. Each shift register is loaded with a set of initial conditions - these initial conditions determine the actual code to be generated. In order to generate a number of different codes with the same hardware, all that is required is a change in the contents of one of the initial shift registers; the other does not need to be adjusted. The configuration of the hardware remains the same, and does not need to be altered. Hence, we can produce a large number of Gold codes with a simple piece of fixed hardware, and a variable input to one of the shift registers.

The choice of initial conditions to one of the shift registers is very important. It is possible that any random choice of bits will produce a PN code, but it is highly likely that such a code will not be balanced, and will not possess an optimum cross correlation value. In order to produce an efficient code, a fixed procedure must be followed. Such a procedure is complicated, and involves the long division of polynomials. Once such a division has been performed, the calculation of optimum cross correlation codes then follows relatively easily. By shifting one of the sequences past the other, and modulo-2 adding them together, another Gold code is produced. This Gold code may or may not, be balanced, but since balanced codes are preferable, it is important to choose the original code combination carefully.

Bekir [37] used moments to establish the upper and lower bounds on the prob-

ability functions for cross correlation values of sets of Gold sequences. Weber *et al.* [38] was then able to conclude that for a set of Gold sequences of degree 13, the effect of other users in a CDMA network can effectively be modeled as a Gaussian random variable at the receiver. To this end, an analysis of a 13 stage Gold code generator shall now be performed.

Following the Gold–derived algorithm outlined by Dixon [39], a pair of preferred polynomials was chosen to give the required cross correlation properties. Peterson and Weldon [40] present a table of irreducible polynomials up to degree 34. Choosing the initial polynomial of degree 13 as 20033, it follows from Dixon [39] that the second polynomial can be calculated using the relation $2^{\frac{(n-1)}{2}} + 1 = 65$ where n is the degree of the sequence. Using the tables from Peterson and Weldon [40], we find that for a sequence of degree 13, with polynomial root 65, our required polynomial becomes 33343. Hence our two polynomials are

$$20033 \rightarrow 010\ 000\ 000\ 011\ 011$$

ie,

$$f(x) = x^{13} + x^4 + x^3 + x + 1 \tag{1.15}$$

and,

$$33343 \rightarrow 011\ 011\ 011\ 100\ 011$$

ie,

$$g(x) = x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^6 + x^5 + x + 1 \tag{1.16}$$

The characteristic sequence generated by the shift register corresponding to polynomial 20033 is given by $h(x)/f(x)$ where

$$h(x) = \frac{d(x \cdot f(x))}{dx} \tag{1.17}$$

with co–efficients interpreted mod 2 ie,

$$\frac{\frac{d}{dx}(x + x^2 + x^4 + x^5 + x^{14}) \bmod 2}{1 + x + x^3 + x^4 + x^{13}} \tag{1.18}$$

now the numerator is given by

$$(1 + 2x + 4x^3 + 5x^4 + 14x^{13}) \bmod 2 = 1 + x^4 \tag{1.19}$$

thus the characteristic phase (the initial conditions) can be found from the quotient

$$\frac{1 + x^4}{1 + x + x^3 + x^4 + x^{13}}$$

$$= 1 + x + x^2 + 0x^3 + x^4 + x^5 + 0x^6 + x^7 + x^8 + 0x^9 + x^{10} + x^{11} + 0x^{12} + \ldots \tag{1.20}$$

and is calculated as follows:

$$
\begin{array}{r}
1+x+x^2+0x^3+x^4+x^5+0x^6+x^7+x^8+0x^9+x^{10}+x^{11}+0x^{12}+\ldots \\
1+x+x^3+x^4+x^{13}\;\overline{)\;1+x^4} \\
\underline{1+x+x^3+x^4+x^{13}} \\
x+x^3+x^{13} \\
\underline{x+x^2+x^4+x^5+x^{14}} \\
x^2+x^3+x^4+x^5+x^{13}+x^{14} \\
\underline{x^2+x^3+x^5+x^6+x^{15}} \\
x^4+x^6+x^{13}+x^{14}+x^{15} \\
\underline{x^4+x^5+x^7+x^8+x^{17}} \\
x^5+x^6+x^7+x^8+x^{13}+x^{14}+x^{15}+x^{17} \\
\underline{x^5+x^6+x^8+x^9+x^{18}} \\
x^7+x^9+x^{13}+x^{14}+x^{15}+x^{17}+x^{18} \\
\underline{x^7+x^8+x^{10}+x^{11}+x^{20}} \\
x^8+x^9+x^{10}+x^{11}+x^{13}+x^{14}+x^{15}+x^{17}+x^{18}+x^{20} \\
\underline{x^8+x^9+x^{11}+x^{12}+x^{21}} \\
x^{10}+x^{12}+x^{13}+x^{14}+x^{15}+x^{17}+x^{18}+x^{20}+x^{21} \\
\underline{x^{10}+x^{11}+x^{13}+x^{14}+x^{23}} \\
x^{11}+x^{12}+x^{15}+x^{17}+x^{18}+x^{20}+x^{21}+x^{23} \\
\underline{x^{11}+x^{12}+x^{14}+x^{15}+x^{24}} \\
\vdots
\end{array}
$$

Hence, the resulting initial conditions for the shift register are 1110110110110.

These initial conditions determine the power–on state of the lower shift register, and do not need to be altered to produce different Gold codes. In order to generate an alternative Gold code, the initial conditions of the upper shift register are loaded with different values. The only restriction on these values is that the first stage (the one on the right) contain a zero. If this is not the case, an unbalanced code will be produced and as a result will possess a non–zero DC offset. The structure of the resulting Gold code generator is shown in Figure 1.10, where the lower shift register has a fixed set of initial conditions, and the upper register can be dynamically configured.

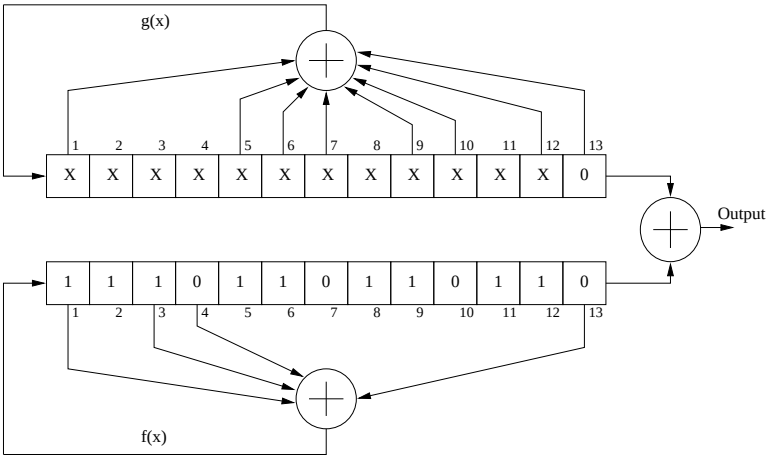

Figure 1.10: 13 stage Gold code generator

1.4.1 Kasami Sequences

Following a procedure similar to that used to produce Gold codes will generate a smaller set of binary sequences known as Kasami sequences. A set of Kasami sequences consists of $M = 2^{m/2}$ binary sequences with a period of $n = 2^m - 1$, provided that m is even. Starting with an $m-$sequence, say A, a second sequence, B, can be obtained by taking every $2^{m/2} + 1$ bit of A, or in other words, sequence B is generated by decimating A by $2^{m/2} + 1$. It can be shown that the resulting sequence is periodic with period $2^{m/2} - 1$. By taking $n = 2^m - 1$ bits of the sequences A and B, a new set of sequences is formed by modulo-2 adding the bits from A and the bits from B, and all the $2^{m/2} - 2$ cyclic shifts of the bits from B. By including A in the set, a set of $2^{m/2}$ binary sequences of length $n = 2^m - 1$ is obtained. The auto and cross correlation functions of Kasami sequences can assume values from the set $\{-1, -(2^{m/2} + 1), 2^{m/2} - 1\}$. Hence, the maximum value of cross correlation for any pair of sequences from the set is $2^{m/2} + 1$.

1.5 Error Correcting Codes

Error correcting codes are finding an increase in acceptance in the spread spectrum community. A code which is able to dynamically decide if it has suffered from interference, mark the appropriate bits, and then attempt to correct the problem is certainly of value. This section will introduce block codes, convolution coding, the Viterbi algorithm and Reed-Solomon coding, all of which are becoming more prevalent in today's spread spectrum technology.

In conventional narrow band communications systems, the use of forward error correction requires either that the transmission bandwidth be increased or that the information data rate be decreased to account for the redundancy of the code. In contrast to this, the use of forward error correction in a spread spectrum system does not require an increase in bandwidth or a reduction in the data rate. Furthermore, Viterbi [7] was able to show that the magnitude of the processing gain does not decrease when forward error correction is applied.

It is assumed that interleaving will be used when applying a method of forward error correction, since this technique performs best when channel errors are independent from one transmission interval to the next. The role of the interleaver is to rearrange the order in which the transmitted data symbols are sent, so that bursts of noise in the channel will not appear as bursts at the decoder since they are reordered at the receiver using a de-interleaving scheme. This has the effect

of reducing a stream of what would otherwise be error bits, into a form of AWGN which is more easily dealt with.

1.5.1 Block Codes

Consider a coding scheme which outputs an $n-$bit binary codeword for each group of k input binary data bits. Each group of k encoder input signals is called an input message, and shall be identified by a message number, m. There are 2^k possible input words, with each possessing a unique output codeword. The code rate, R, is given by $R = k/n$.

Given that the error correcting capability of a code is due to the fact that not all code combinations will be used, it is possible to generate codes such that a number of transmission errors must occur before one codeword could be confused with another.

Codewords are represented by binary $n-$tuples, $\mathbf{x}_m = (x_{m1}, x_{m2}, \ldots, x_{mn})$, where $m = 0, 1, 2, \ldots, 2^k - 1$ is the message associated with the codeword. Any two codewords $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ which differ from one another in d_H places and agree in $n-d_H$ places are said to be separated by Hamming distance, d_H. For a binary symmetric channel, a total of d_H transmission errors must occur before $\mathbf{x}_m$ is changed into $\mathbf{x}_{m'}$. The minimum Hamming distance between any two of the 2^k codewords in a code is known as the minimum distance, d_{min} of that code. An (n, k) binary block code uses a fraction $2^k/2^n = 2^{k-n}$ of all possible output codewords. A low-rate code uses a smaller fraction of the possible output words than a high rate code and codewords can therefore be separated further from one another. Thus low rate codes typically have more error correction capability than high-rate codes.

An important property of linear block codes is that the modulo-2 addition of any two codewords results in another codeword. Consider two codewords $\mathbf{x}_a$ and $\mathbf{x}_b$ separated by a Hamming distance d_H, and a third arbitrary codeword $\mathbf{x}_c$. Let $\mathbf{x}_{a'} = \mathbf{x}_a \oplus \mathbf{x}_c$ and $\mathbf{x}_{b'} = \mathbf{x}_b \oplus \mathbf{x}_c$, and notice that the Hamming distance between $\mathbf{x}_{a'}, \mathbf{x}_{b'}$ is also d_H. Using these two properties, it can be demonstrated that the set of Hamming distances between a codeword $\mathbf{x}_m$ and all other codewords $\mathbf{x}_{m'}, m' \neq m$, is the same for all m. If the number of codewords which are Hamming distance, $d_H = d$ from the all zeros codeword is denoted by A_d, then the weight distribution of the code is the set of all A_d for $d = d_{min}, \ldots, n$.

Consider the binary block code defined in Table 1.3. This code uses $k = 4, n = 7$, and has rate $R = 4/7$ and minimum distance 2. There are a total of $2^k = 16$ codewords. The weight distribution of this code is $A_3 = 5, A_4 = 6$, and $A_7 = 1$. Thus there are a total of five codewords $\mathbf{x}_{m'}$ with distance 3, six codewords $\mathbf{x}_{m'}$ with

distance 4, and one codeword $\mathbf{x}_{m'}$, with distance 7 from any particular codeword $\mathbf{x}_m$.

Table 1.3: A typical binary block code

Message Number	Message	Codeword
0	0 0 0 0	0 0 0 0 0 0 0
1	0 0 0 1	1 0 1 0 0 0 1
2	0 0 1 0	1 1 0 0 0 1 0
3	0 0 1 1	0 1 1 0 0 1 1
4	0 1 0 0	1 1 1 0 1 0 0
5	0 1 0 1	0 0 0 0 1 0 1
6	0 1 1 0	0 0 1 0 1 1 0
7	0 1 1 1	1 0 1 0 1 1 1
8	1 0 0 0	1 1 0 1 0 0 0
9	1 0 0 1	1 0 1 1 0 0 1
10	1 0 1 0	1 1 0 1 0 1 0
11	1 0 1 1	1 0 0 1 0 1 1
12	1 1 0 0	0 0 1 1 1 0 0
13	1 1 0 1	0 1 0 1 1 0 1
14	1 1 1 0	0 1 1 1 1 1 0
15	1 1 1 1	1 1 1 1 1 1 1

1.5.2 Convolutional Codes

Convolutional codes differ from block codes in that bit streams of data are not grouped into distinct independent blocks and encoded, but rather, a continuous sequence of data bits is mapped into a continuous sequence of encoder output symbols. The mapping of input data streams into encoder output sequences is done in such a manner that communication efficiency is improved by enabling the system to correct for transmission errors.

Convolutional codes use the same optimum decoding rule as that used for block codes. That is, the decoder should estimate the transmitted code sequence, given the known code structure, channel characteristics, and received sequence. The sequence most likely to have been transmitted is that which is closest in Hamming distance to the received sequence.

A two stage shift register circuit that generates a simple convolutional code with a rate of 0.5 is shown in Figure 1.11. After each input is clocked in, the output sequence is generated by sampling and multiplexing the outputs of the two modulo-2 adders. With this simple sequence, two output symbols ($n = 2$) are generated for

each input bit ($k = 1$) and the code rate is $R = k/n = \frac{1}{2}$. Note that a particular bit influences the output sequence during its own clocking interval as well as the next two input clock cycles. A convolutional code is defined by the number of stages in the shift register, the number of outputs, and the feedback connections between the shift register and the modulo-2 adders. The state of the encoder is defined to be the contents of the shift register and is completely determined by the previous two information bit inputs. The convolutional encoder shown in Figure 1.11 has four possible states corresponding to all possible combinations of the two stage shift register.

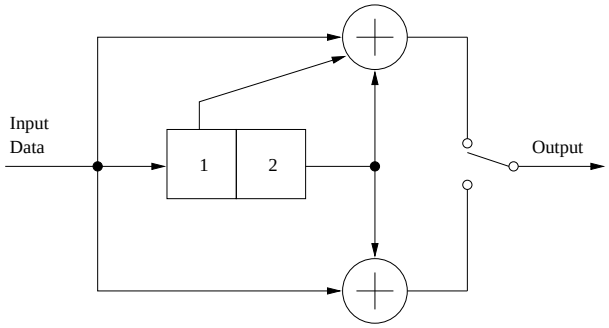

Figure 1.11: Convolutional encoder with a rate of 0.5

The convolutional code just described has a code rate of $R = 0.5$, and produces two output bits for each input bit. In general, a convolutional code produces n output symbols for every k input symbols and has a code rate of $R = k/n$. Although this would appear to make convolutional codes similar to block codes, they are in fact significantly different. Most importantly, convolutional codes are said to have memory, meaning the mapping function used to map the k information bits into n code symbols is a function of the previous data bits. For example, the encoder of Figure 1.11 maps one data bit into two code symbols using a function that relies on the previous two data bits. For example, an input 1 produces outputs 11, 00, 01 or 10 when the previous two inputs, and hence the encoder states are 00, 10, 01, or 11 respectively. The number of past inputs that affect the mapping of the current inputs is a critical parameter of convolutional codes. This parameter affects convolutional code performance and complexity in a similar way that block length affects the operation of block codes. An indication of the amount of memory within the decoder is known as the constraint length. The constraint length of the code is 1 plus the number of past inputs affecting the current outputs. The constraint length of the code of Figure 1.11 is 3, since the current output pair is a function of the current input plus the previous two inputs.

The convolutional code of Figure 1.11 can also be represented by a state tran-

sition diagram, as shown in Figure 1.12. The state of the encoder is represented by the contents of the circles, and starts in state 00. If the first input bit is a 0, the encoder exits state 00 on the solid branch, outputs the two code symbols found along this branch (00), and returns to state 00. However, if the first input bit was a 1, the encoder would exit state 00 on the dashed branch, would output the data bits 11, and would enter state 10.

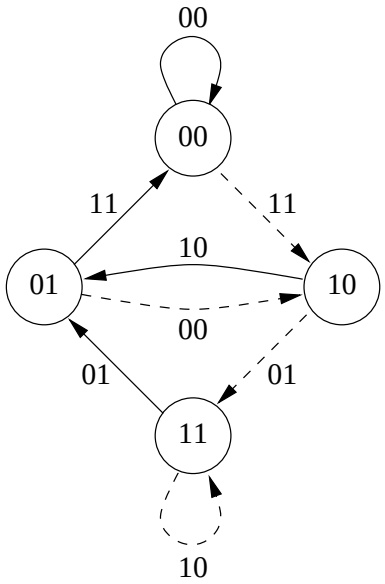

Figure 1.12: State transition diagram for rate 0.5 convolutional encoder

In general, input bits equal to 0 cause the encoder to move along the solid branches to the next state. outputting the codeword symbols corresponding to the branch label, whilst encoder inputs equal to 1 cause the encoder to move along the dashed branches. Convolutional coders can therefore be considered as finite state machines that change state as a function of the input sequence. The branch labels in Figure 1.12 were calculated directly from the shift register representation shown in Figure 1.11.

The convolutional coder shown in Figure 1.11 can also be represented by a trellis diagram as illustrated in Figure 1.13. This encoder has four states, represented by the circles, which correspond to the states of the state transition diagram. The encoding starts at state 00 on the far left of the trellis. If the first information bit is a 0, the encoder moves along the solid line out of state 00, arriving again at state 00. The encoder output is the symbol pair 00. If the first encoder input was a 1, the encoder would move along the dashed branch out of state 00, arriving at state 10, and would output the sequence 11.

In order to decode a convolutional code, the decoder must have knowledge of the code structure (the code trellis), the received sequence, and a statistical characteri-

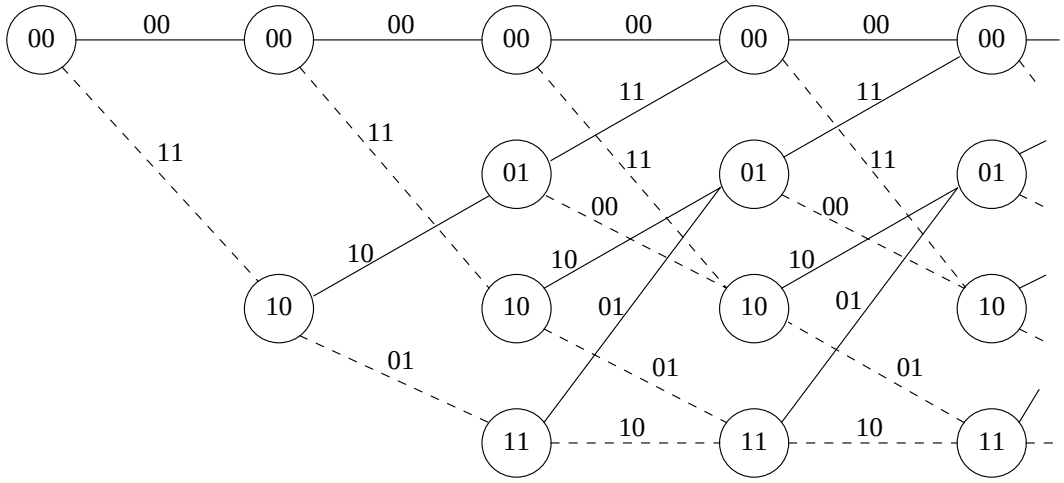

Figure 1.13: Trellis diagram for a convolutional encoder with rate 0.5

zation of the channel. The function of the decoder is to estimate the encoder input information sequence using a rule or method that results in the minimum possible number of errors being delivered to the user. There is a one to one correspondence between information sequences and encoder output sequences. Further, any information and code sequence pair is uniquely associated with a path through the trellis. Thus the job of the decoder may be viewed as estimating the path through the trellis which was followed by the coder.

When decoding convolutional codes, it is assumed that the information source is ideal so that its output symbols are equally likely and independent; thus there is an equal probability for all paths through the trellis and a maximum likelihood decoding rule can be used. The decoder output is the path whose code sequence was most likely to have been transmitted by the encoder. Denote the message sequence m, the encoder output sequence, by $\mathbf{x}_m = x_{m0}x_{m1}x_{m2}\ldots x_{mj}\ldots$. The transmitted sequence can denoted by $\mathbf{y} = y_0y_1y_2\ldots y_j\ldots$, and the probability of receiving the transmitted sequence $\mathbf{y}$ given that the transmitted sequence was $\mathbf{x}_m$ is

$$p(\mathbf{y}|\mathbf{x}_m) = \prod_{j=0}^{\infty} p(y_j|x_{mj}) \tag{1.21}$$

It is convenient for the decoder to use the term $\ln\{p(\mathbf{y}|\mathbf{x}_m)\}$ instead of $p(\mathbf{y}|\mathbf{x}_m)$, since the logarithm is a monotone increasing function of an increasing argument [2]. In this case the decoder would calculate $\ln\{p(\mathbf{y}|\mathbf{x}_m)\}$ for all paths and would choose the path m with the largest value as the decoder output path. Taking the logarithm of 1.21 converts the product to a summation. Thus the decoding formula may be stated as:

$$\ln\{p(\mathbf{y}|\mathbf{x}_m)\} = \sum_{j=1}^{\infty} \ln\{p(y_j|x_{mj})\} \tag{1.22}$$

This is equivalent to finding the path through the trellis whose code sequence is closest in Hamming distance to the received sequence. A common method for finding this closest path is the Viterbi algorithm.

1.5.3 The Viterbi Algorithm

The Viterbi algorithm is a commonly used method for performing maximum likelihood decoding of convolutional codes. Because of the structure of the codes, convolutional decoders can make use of channel output information. To create a hard-decision channel, the output values are partitioned into two regions, corresponding to either a 0 or 1. A soft-decision decoder quantizes this space into more than two discrete values, but regardless of whether the channel outputs hard or soft decisions, the decoding rule remains the same. Given the received sequence $\mathbf{y}$, the decoder output is the path, or code sequence $\mathbf{x}_m$, that maximises the probability $p(\mathbf{y}|\mathbf{x}_m)$.

Hard-Decision Decoding

Consider the trellis diagram shown in Figure 1.14, and assume a received sequence of 10 10 10 11 11 10 01 11. The initial decoding task is to calculate the Hamming distance between the first two received symbols and those on the trellis at depth 0 leading to the depth 1 states. These Hamming distances are both 1, since there is one bit of difference between the received sequence, and the trellis code bits. The decoder stores these Hamming distances and moves to depth 2 of the trellis. Here the decoder calculates the Hamming distance between the first four received symbols and the first four symbols on all four trellis paths leading into the depth 2 states. The Hamming distance at a a depth g, along any path is the sum of the previous Hamming distances along that path, and the distance accumulated in the branch leading from depth $g-1$ to depth g. Thus the decoder only needs to add the Hamming distance it stored in the previous step to the most recently accumulated Hamming distance. The Hamming distances at each depth are shown on the trellis diagram in Figure 1.15.

At depth three, there are two trellis paths leading into each of four possible states. Therefore, eight path segments must be considered. After summing the accumulated Hamming distance along each path segment, the decoder is able to discard one of the two paths leading into each state. The path having the largest

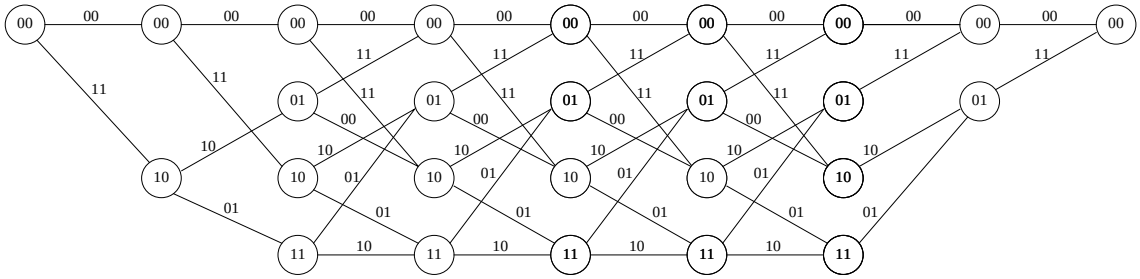

Figure 1.14: Trellis diagram for Viterbi Decoding

accumulated distance may be discarded, since it has merged with a path having a lower cumulative Hamming distance, so it can never form a path through the trellis having the lowest total distance. This discarding of one of the two paths leading into each state is the key to the efficiency of the Viterbi algorithm.

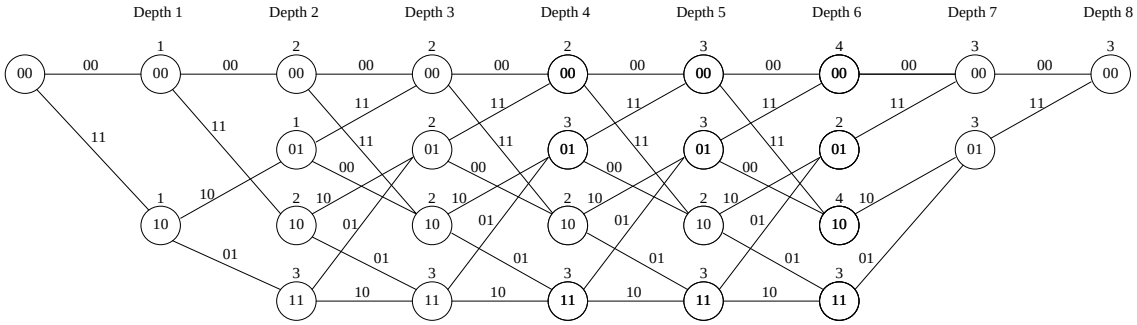

Figure 1.15: Trellis diagram showing Hamming distances

Soft-Decision Decoding

Viterbi decoding using soft decision works in exactly the same way as the Hamming distance method. A minor difference is that for soft-decision decoding, the path with the largest log-likelihood value, $\ln\{p(\mathbf{y}|\mathbf{x}_m)\}$ is being found rather than the path with the smallest Hamming distance. Thus, the segments which are discarded are those with the smaller of the two values. In both cases the path most likely to have been followed by the encoder is found.

1.5.4 Threshold Decoding

There are certain convolutional codes which can be decoded using relatively simple hardware. In order to be decoded so simply, the codes require additional structure as described by Massey [5] and Costello [4]. The simplicity of threshold detectors enables their implementation at very high speeds. Although the performance of these special convolutional codes is not outstanding for use under the influence of

AWGN, their structure is particularly convenient for applications where interleaving is required. For channels where interference is likely to arrive in bursts, very large coding gains are possible by making use of convolutional coding with threshold decoding [2].

1.5.5 Reed-Solomon Codes

Reed-Solomon codes have long been used in the communications industry in an attempt to produce a system which is immune to interference. More recently though, this coding scheme has found favour with the audio community, and is now is common use amongst developers of compact disk equipment.

Let b be the symbol size in bits, and $RS(N, K)$ denote a Reed-Solomon symbol code with length N occupying a vector space of dimension K. This code will exist whenever $0 < K < N \leq 2^b + 1$. It should be noted that the parameters N, K, and b do not fully specify a Reed-Solomon code, and many trade-offs and incompatibilities exist within these parameters. Assuming that $N = 2^b - 1$, the encoding process can be specified as follows.

Let $a_0, a_1, \ldots, a_{K-1}$ be elements representing the information to be encoded, thus forming a polynomial $a(x)$ with a maximum degree of $K - 1$,

$$a(x) = \sum_{i=0}^{K-1} a_i x^i \tag{1.23}$$

The polynomial is then evaluated, leading to the $2^b - 1$ elements that by definition form the code word c. With respect to the minimum distance, note that the sum of two polynomials of maximum degree $K - 1$ is also a polynomial with maximum degree of $K - 1$. In other words, the code produced is a linear one. It is worth noting that Reed-Solomon codes meet the Singleton bound of $d = N - K + 1$ [41]. Codes meeting this bound are known as maximal distance separable (MDS). Although it is not true that an MDS code is necessarily a Reed-Solomon code, it is true that all known MDS codes have the same properties as Reed-Solomon codes. From this point of view, Reed-Solomon codes are an important class of optimal codes.

Reed-Solomon codes are more suited to high speed implementation than comparable binary codes. As an example, consider the RS(255,223) code which takes 223 bytes of data and encodes them into 255 bytes. Although this may appear to be wasting bandwidth, the decoder and encoding logic can be designed to use byte-based arithmetic throughout. The decoder then processes 255 objects of data rather than 255 sets of 8 bit binary data, thus reducing the overall complexity of the logic. In addition, data transfers can be made on a byte-wide basis; a 50 MHz

Reed-Solomon decoder is operationally equivalent to a $50 \times 8 = 400$ Mbit/s binary decoder.

The rate of decoding errors when using Reed-Solomon codes is very low. If the number of symbol errors exceeds the amount that the code can correct, the decoder will primarily recognise the impossibility of decoding and flag the data as corrupt. The error rate of data that is both undecodable and not recognised as such by the decoder may typically be as low as five orders of magnitude below the overall symbol error rate [42].

1.6 Conclusion

This chapter has discussed the concept of pseudo-random sequences as applicable to spread spectrum communications. Maximal length sequences were introduced, and used as an introduction to more complicated methods of PN code generation. Gold codes were examined, and it was shown that for a CDMA network, a code length of 2^{13} bits is required for effective operation. Auto and cross correlation were presented using both theoretical and numerical approaches, and the requirements for appropriate coding schemes were related to their relevant correlation properties. A brief examination of Kasami sequences was made, and error correcting codes were examined. Block coding, convolutional codes, the Viterbi algorithm, and Reed-Solomon coding schemes were discussed.

Bibliography

[1] D. L. Nicholson, *Spread Spectrum Signal Design: LPE and AJ Systems* Rockville, MD, Computer Science Press, 1988

[2] R. L. Peterson, R. E. Ziemer and D. E. Borth, *Introduction to Spread Spectrum Communications*, Upper Saddle River, NJ, Prentice Hall, 1995

[3] R. E. Ziemer and W. H. Tranter, *Principles of Communications: Systems, Modulation, and Noise* 4th ed., Boston, Houghton Mifflin, 1995

[4] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, NJ, Prentice Hall, 1983

[5] J. L. Massey, *Threshold Decoding*, Cambridge, Mass., MIT Press, 1963

[6] R. L. Picholtz, D. L. Schilling, and L. B. Milstein, "Theory of spread-spectrum communications - A tutorial", *IEEE Trans. Commun.*, vol. COM-30, pp. 855-884, May 1982.

[7] A. J. Viterbi, "Spread spectrum communications - Myths and realities", *IEEE Commun. Mag.*, pp. 11-18, May 1979.

[8] M. Simon, J. Omura, R. Scholtz, and K. Levitt, *Spread Spectrum Communications*, Vols. I, II, and III. Rockville, MD, Computer Science Press, 1985

[9] R.C. Dixon, *Spread Spectrum Systems*, 2nd Ed. John Wiley and Sons, New York, pp. 155-157, 1984.

[10] L. B. Milstein, "Interference rejection techniques in spread-spectrum communications", *Proc. IEEE*, vol. 76, pp. 657-671, June 1988.

[11] R.C. Dixon, *Spread Spectrum Systems*, 2nd Ed. John Wiley and Sons, New York, pp. 214-248, 1984.

[12] W. C. Lindsey and J. L. Lewis, "Modeling, characterization and measurement of oscillator frequency instability", *NRL Report*, NRL-1.1351, June 14, 1974.

[13] R. Ward, "Acquisition of pseudonoise signals by sequential estimation", *IEEE Trans. Commun. Tech.*, pp. 475-483, Dec. 1965.

[14] D. R. Anderson, "A new class of cyclic codes", *SIAM J. Appl. Math.*, 16, No. 1, 1968.

[15] J. J. Spilker, "GPS signal structure and performance characteristics", *Navigation: J. Inst. Navigation*, vol. 25, no. 2, pp. 121-146, 1978.

[16] G. L. Turin, "An Introduction to Matched Filters", *IRE Trans. Info. Th.*, June 1960.

[17] Y. Lee and S. Tantaratana, "Sequential acquisition of PN sequences for DS/SS communications: Design and performance", *IEEE J. Select. Commun.*, vol 10. pp. 750-759, May 1992.

[18] A. K. Elhakeem, G. S. Takhar, and S. C. Gupta, "New code acquisition techniques in spread spectrum communications", *IEEE Trans. Commun.*, vol. COM-28, pp. 246-259, Feb. 1980.

[19] R.C. Dixon, *Spread Spectrum Systems*, 2nd Ed. John Wiley and Sons, New York, pp. 248-260, 1984.

[20] R. Nettleton and H. Alavi, "Power control for a spread spectrum cellular mobile radio system", *Proc. 33rd IEEE Vehic. Tech. Conf.*, Toronto, Canada, pp. 242-246, May, 1983.

[21] G. L. Stuber and C. Kchao, "Analysis of a multiple-cell direct sequence CDMA cellular mobile radio system", *IEEE J. Select. Commun.*, vol. 10, pp. 669-679, May 1992.

[22] F. Simpson and J. Holtzman, "CDMA power control, interleaving, and coding", *Proc. 41st IEEE Vehic. Technol. Conf.*, Saint Louis, MO, pp. 362-367, 1991.

[23] T. Pagden, "Spectrum Spreader", *VHDL RadioComms Library*, Doulos, Hampshire, England, Apr. 1997.

[24] A. Duncan, "A 32x16 Reconfigurable Correlator for the XC6200", *Application Note 084*, ver 1.0, Xilinx Corp. July 1997.

[25] Unknown, "Sine/Cosine", *Product Specification*, CORE Generator v1.5, Xilinx Inc. July 1998.

[26] M. Neeracher, "QUETZALCOATL: Matched Filter for Spread Spectrum Modem", *Internal Report*, Integrated Systems Laboratory, ETH Zurich, May 1994.

[27] R. Zimmermann and M. Neeracher, "SHIVA: Correlator/Demodulator Chip for Direct-Sequence Spread-Spectrum RAKE-Receiver", *Technical report no. 94/9*, Integrated Systems Laboratory, ETH Zurich, May 1994.

[28] S. D. Lingwood, H. Kaufmann and B. Haller, "ASIC Implementation of a Direct-Sequence Spread-Spectrum RAKE-Receiver", *IEEE Vehic. Tech. Conf. VTC'94*, pp. 1326-1330, Jun 1994.

[29] C. Uhl, J. J. Monot and M. Margery, "Single ASIC CDMA Digital Receiver for Space Applications", *IEEE Vehic. Tech. Conf. VTC'94*, pp. 1331-1335, Jun 1994.

[30] S. Chan and V. Leung, "Design and Implementation of a Code-Phase-Shift Keying Spread Spectrum Receiver Employing a FPGA baseband decoder", *IEEE Pacific Rim Conf. on Comm.*, pp. 632-635, Aug. 1997.

[31] J. Kong, Y. Park and M. Kim, "A Fast Serial Viterbi Decoder for CDMA Cellular Base Station.", *IEEE TENCON DSP App. Proc.*, pp. 333-337, Nov. 1996.

[32] C. Paar and M. Rosner, "Comparison of Arithmetic Architectures for Reed-Solomon Decoders in Reconfigurable Hardware.", *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, pp. 219-225, Apr. 1997.

[33] D. J. Torreri, *Principles of secure communications*, 2nd ed., Artech House, pp. 112-117, 1992.

[34] W. J. Judge, "Multiplexing using quasiorthogonal functions", *AIEEE Winter General Mtg.*, Jan. 1962.

[35] R. Gold, "Optimal Binary Sequences for Spread Spectrum Multiplexing", *IEEE Trans. Inform. Theory*, pp. 619-621, Oct 1967.

[36] R. Gold, "Maximal Recursive Sequences with 3-valued Recursive Cross-Correlation Functions", *IEEE Trans. Inform. Theory*, pp. 154-156, Jan. 1968.

[37] N.E. Bekir, "Bounds on the distribution of partial correlation for PN and Gold Sequences", *Ph.D. dissertation*, Dep. Elec. Eng., Univ. of Southern California, Los Angeles, Jan. 1978

[38] C.L. Weber, G.K. Huth, B.H. Batson, "Performance Considerations of Code Division Multiple-Access Systems", *IEEE Trans. Vehic. Tech.*, Vol. VT-30, No. 1, pp. 3-10, Feb 1981.

[39] R.C. Dixon, *Spread Spectrum Systems*, 2nd Ed. John Wiley and Sons, New York, pp. 403-405, 1984.

[40] W. Peterson and E. Weldon, *Error Correcting Codes*, 2nd Ed. MIT Press, 1972, pp.472-491.

[41] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, 1978.

[42] S. Golomb, R. Peile, and R. Sholtz, *Basic Concepts in Information Theory and Coding*, Plenum Press, New York, 1994.

[43] P. Alfke, *The Programmable Logic Data Book*, Xilinx Corp., 1984, p. 9-24

[44] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators", *Application Note*, Xilinx Corp., Aug. 1995

[45] D. L. Perry, *VHDL*, 2nd ed., McGraw Hill, New York, 1994.

[46] Green Mountain Computer Systems, "A Brief VHDL Tutorial", `http://together.net/~thibault/home.htm`, Essex Junction, VT, 1998.

[47] P. J. Ashenden, *The VHDL Cookbook*, Dept. Comp. Sci., Uni. Adelaide, 1st ed., July 1990.

[48] D. Hoffman, D. Leonard, C. Linder, K. Phelps, A. Rodger and J. Wall, *Coding Theory*, Marcel Dekker, New York, 1992.

[49] D. J. Torrieri, "Performance of direct-sequence systems with long pseduonoise sequences", *IEEE J. Select. Commun., vol. 10, pp. 770-781, May 1992.*

[50] R. S. Lunayach, "Performance of a direct sequence spread-spectrum system with long period and short period code sequences", *IEEE Trans. Commun.*, vol. COM-31, pp. 412-419, Mar. 1983.

[51] E. R. Berlekamp, "The technology of error-correcting codes", *Proc. IEEE*, vol. 68, May 1980.

[52] S. Fredricsson, "Pseudo-randomness properties of binary shift register sequences", *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 115-120, Jan. 1975.

[53] R. Kohno and H. Imai, "On pseduo-noise sequences for direct-sequence QPSK spread spectrum communication systems", *Trans. IECE Japan*, vol. J65-A, no. 1, pp. 69-76, Jan. 1982.

[54] J. L. Massey, "Coding and modulation in digital communications", *Proc. Int. Zurich Sem. Digital Commun.*, Zurich, Switzerland, 1974, pp. E2(1)-E2(4).

[55] S. El-Khamy and A. Balameshi, "Selection of Gold and Kasami code sets for spread spectrum CDMA systems of limited number of users", *Int. J. Satellite Commun.*, vol. 5, pp. 23-32, Jan.-Mar. 1987.

[56] E. A. Geraniotis, "Performance of noncoherent direct-sequence spread-spectrum multiple-access communications", Special Issue on Military Communications, *IEEE J. Select. Areas Commun.*, vol. SAC-3, pp. 687-694, Sept. 1985.

[57] M. B. Pursley, "Performance evaluation for phase-coded spread spectrum multiple-access communications - Part I: System analysis", *IEEE Trans. Commun.*, vol. COM-25, pp. 795-799, Aug. 1977.

[58] Jenkins, J. *Designing with FPGAs and CPLDs*, Prentice Hall, USA, 1994

[59] A. B. Glenn, "Low probability of intercept", *IEEE Commun. Mag.*, vol. 21, pp.26-33, July 1983.

[60] E. S. Sousa, "The effet of Clock and carrier frequency offsets on the performance of a direct sequence spread spectrum multiple access system", *IEEE J. Select. Areas Commun.*, vol. 8, pp. 580-587, May 1990.

[61] P. A. Kullstam, "Spread spectrum performance analysis in arbitrary interference", *IEEE Trans. Commun.*, vol. COM-25, pp. 848-853, Aug. 1977.

[62] M. Kavehrad and P. J. McLane, "Spread spectrum for indoor digital radio", *IEEE Commun. Mag.*, vol. 25, pp. 32-40, June 1987.

[63] M. Kavehrad and G. E. Bodeep, "Design and experimental results for a direct sequence spread spectrum radio using DPSK modulation for indoor, wireless communications", *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 815-23, June 1987.

[64] R. L. Pickholtz, L.B. Milstein and D. L. Schilling, "Spread spectrum for mobile communications", *IEEE Trans. Vehic. Technol.*, vol. 40, pp. 313-322, May 1991.

[65] H. Meyr and G. Poltzer, "Performance analysis for general PN-spread spectrum acquisition techniques", *IEEE Trans. Commun.*, vol. COM-31, pp. 1317-1319, Dec. 1983.

[66] W. A. Gardner, "The role of spectral correlation in the design and analysis of synchronizers", *IEEE Trans. Commun.*, vol. COM-34, pp. 1089-1095, Nov. 1986.

[67] S. Pan, D. Doods, and S. Kumar, "Acquisition time distribution for spread-spectrum receivers", *IEEE J. Select. Areas Commun.* vol. 8, pp. 800-807, June 1990.

[68] P. M. Hopkins, "A Unified Analysis of Pseudonoise Synchronization by Envelope Correlation", *IEEE Trans. Commun.*, vol. COM-25, pp. 770-778, Aug. 1977.

[69] Xilinx Corp., *HDL Synthesis for FPGAs Design Guide*, 1995, ISBN 040129401.

[70] S. K. Knapp, "Accelerate FPGA Macros with One-Hot Approach", *Electronic Design*, Penton Pub., Sept. 1990.

[71] Accolade Design Automation, *User's Guide to Accolade PeakVHDL*, Prof. ed., Duvall, WA, 1998.