WEEK 11

From Fourier transform theory we know that periodicity in the time domain is transformed into uniform sampling in the frequency domain. This interplay between the time and frequency domains is borne out by the power spectral density of the maximal-length wave $c(t)$. Specifically, taking the Fourier transform of Equation (7.5), we get the sampled spectrum

$$S_c(f) = \frac{1}{N^2}\,\delta(f) + \frac{1+N}{N^2} \sum_{\substack{n=-\infty \\ n \neq 0}}^{\infty} \operatorname{sinc}^2\left(\frac{n}{N}\right) \delta\left(f - \frac{n}{NT_c}\right) \tag{7.6}$$

which is plotted in Figure 7.3c for $m = 3$ or $N = 7$.

Comparing the results of Figure 7.3 for a maximal-length sequence with the corresponding results shown in Figure 1.11 for a random binary sequence, we may make the following observations:

- For a period of the maximal-length sequence, the autocorrelation function $R_c(\tau)$ is somewhat similar to that of a random binary wave.
- The waveforms of both sequences have the same envelope, $\operatorname{sinc}^2(fT)$, for their power spectral densities. The fundamental difference between them is that whereas the random binary sequence has a continuous spectral density characteristic, the corresponding characteristic of a maximal-length sequence consists of delta functions spaced $1/NT_c$ Hz apart.

As the shift-register length m, or equivalently, the period N of the maximal-length sequence is increased, the maximal-length sequence becomes increasingly similar to the random binary sequence. Indeed, in the limit, the two sequences become identical when N is made infinitely large. However, the price paid for making N large is an increasing storage requirement, which imposes a practical limit on how large N can actually be made.

▪ CHOOSING A MAXIMAL-LENGTH SEQUENCE

Now that we understand the properties of a maximal-length sequence and the fact that we can generate it using a linear feedback shift register, the key question that we need to address is: How do we find the feedback logic for a desired period N? The answer to this

TABLE 7.1 ***Maximal-length sequences of shift-register lengths 2–8***

Shift-Register Length, m	*Feedback Taps*
2*	[2, 1]
3*	[3, 1]
4	[4, 1]
5*	[5, 2], [5, 4, 3, 2], [5, 4, 2, 1]
6	[6, 1], [6, 5, 2, 1], [6, 5, 3, 2]
7*	[7, 1], [7, 3], [7, 3, 2, 1], [7, 4, 3, 2], [7, 6, 4, 2], [7, 6, 3, 1], [7, 6, 5, 2], [7, 6, 5, 4, 2, 1], [7, 5, 4, 3, 2, 1]
8	[8, 4, 3, 2], [8, 6, 5, 3], [8, 6, 5, 2], [8, 5, 3, 1], [8, 6, 5, 1], [8, 7, 6, 1], [8, 7, 6, 5, 2, 1], [8, 6, 4, 3, 2, 1]

question is to be found in the theory of error-control codes, which is covered in Chapter 10. The task of finding the required feedback logic is made particularly easy for us by virtue of the extensive tables of the necessary feedback connections for varying shift-register lengths that have been compiled in the literature. In Table 7.1, we present the sets of maximal (feedback) taps pertaining to shift-register lengths $m = 2, 3, \ldots, 8$.[3] Note that as m increases, the number of alternative schemes (codes) is enlarged. Also, for every set of feedback connections shown in this table, there is an "image" set that generates an identical maximal-length code, reversed in time sequence.

The particular sets identified with an asterisk in Table 7.1 correspond to *Mersenne prime length sequences*, for which the period N is a prime number.

▶ EXAMPLE 7.2

Consider a maximal-length sequence requiring the use of a linear feedback-shift register of length $m = 5$. For feedback taps, we select the set [5, 2] from Table 7.1. The corresponding configuration of the code generator is shown in Figure 7.4*a*. Assuming that the initial state is 10000, the evolution of one period of the maximal-length sequence generated by this scheme is shown in Table 7.2*a*, where we see that the generator returns to the initial 10000 after 31 iterations; that is, the period is 31, which agrees with the value obtained from Equation (7.2).

Suppose next we select another set of feedback taps from Table 7.1, namely, [5, 4, 2, 1]. The corresponding code generator is thus as shown in Figure 7.4*b*. For the initial state 10000, we now find that the evolution of the maximal-length sequence is as shown in Table 7.2*b*. Here again, the generator returns to the initial state 10000 after 31 iterations, and so it should. But the maximal-length sequence generated is different from that shown in Table 7.2*a*.

Clearly, the code generator of Figure 7.4*a* has an advantage over that of Figure 7.4*b*, as it requires fewer feedback connections. ◀

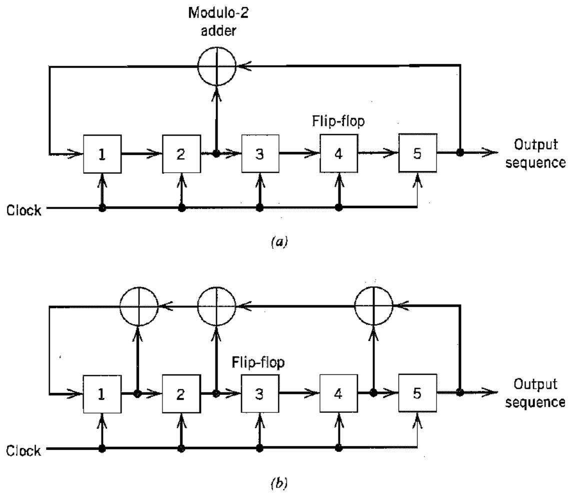

FIGURE 7.4 Two different configurations of feedback shift register of length $m = 5$. (*a*) Feedback connections [5, 2]. (*b*) Feedback connections [5, 4, 2, 1].

TABLE 7.2a ***Evolution of the maximal-length sequence generated by the feedback-shift register of Fig. 7.4a***

Feedback Symbol	*State of Shift Register*					*Output Symbol*
	1	0	0	0	0	
0	0	1	0	0	0	0
1	1	0	1	0	0	0
0	0	1	0	1	0	0
1	1	0	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	0	1	0
0	0	1	1	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1
0	0	1	1	0	1	1
0	0	0	1	1	0	1
0	0	0	0	1	1	0
1	1	0	0	0	1	1
1	1	1	0	0	0	1
1	1	1	1	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0
0	0	1	1	1	1	1
0	0	0	1	1	1	1
1	1	0	0	1	1	1
1	1	1	0	0	1	1
0	0	1	1	0	0	1
1	1	0	1	1	0	0
0	0	1	0	1	1	0
0	0	0	1	0	1	1
1	1	0	0	1	0	1
0	0	1	0	0	1	0
0	0	0	1	0	0	1
0	0	0	0	1	0	0
0	0	0	0	0	1	0
1	1	0	0	0	0	1

Code: 0000101011101100011111001101001

TABLE 7.2*b* ***Evolution of the maximal-length sequence generated by the feedback-shift register of Fig. 7.4b***

Feedback Symbol	*State of Shift Register*					*Output Symbol*
	1	0	0	0	0	
1	1	1	0	0	0	0
0	0	1	1	0	0	0
1	1	0	1	1	0	0
0	0	1	0	1	1	0
1	1	0	1	0	1	1
0	0	1	0	1	0	1
0	0	0	1	0	1	0
1	1	0	0	1	0	1
0	0	1	0	0	1	0
0	0	0	1	0	0	1
0	0	0	0	1	0	0
1	1	0	0	0	1	0
0	0	1	0	0	0	1
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0
1	1	1	1	1	0	1
1	1	1	1	1	1	0
0	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
0	0	1	1	0	1	1
0	0	0	1	1	0	1
1	1	0	0	1	1	0
1	1	1	0	0	1	1
1	1	1	1	0	0	1
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	0	1	1	1
0	0	0	0	0	1	1
1	1	0	0	0	0	1

Code: 0000110101001000101111101100111

7.3 A Notion of Spread Spectrum

An important attribute of spread-spectrum modulation is that it can provide protection against externally generated interfering (jamming) signals with finite power. The jamming signal may consist of a fairly powerful broadband noise or multitone waveform that is directed at the receiver for the purpose of disrupting communications. Protection against jamming waveforms is provided by purposely making the information-bearing signal occupy a bandwidth far in excess of the minimum bandwidth necessary to transmit it. This has the effect of making the transmitted signal assume a noiselike appearance so as to blend into the background. The transmitted signal is thus enabled to propagate through the channel undetected by anyone who may be listening. We may therefore think of spread spectrum as a method of "camouflaging" the information-bearing signal.

One method of widening the bandwidth of an information-bearing (data) sequence involves the use of *modulation*. Let $\{b_k\}$ denote a binary data sequence, and $\{c_k\}$ denote a pseudo-noise (PN) sequence. Let the waveforms $b(t)$ and $c(t)$ denote their respective polar nonreturn-to-zero representations in terms of two levels equal in amplitude and opposite in polarity, namely, ± 1. We will refer to $b(t)$ as the information-bearing (data) signal, and to $c(t)$ as the PN signal. The desired modulation is achieved by applying the data signal $b(t)$ and the PN signal $c(t)$ to a product modulator or multiplier, as in Figure 7.5*a*. We know from Fourier transform theory that multiplication of two signals produces a signal whose spectrum equals the convolution of the spectra of the two component signals. Thus, if the message signal $b(t)$ is narrowband and the PN signal $c(t)$ is wideband, *the product (modulated) signal $m(t)$ will have a spectrum that is nearly the same as the wideband PN signal.* In other words, in the context of our present application, the PN sequence performs the role of a *spreading code.*

By multiplying the information-bearing signal $b(t)$ by the PN signal $c(t)$, each information bit is "chopped" up into a number of small time increments, as illustrated in the waveforms of Figure 7.6. These small time increments are commonly referred to as *chips*.

For *baseband* transmission, the product signal $m(t)$ represents the *transmitted signal.* We may thus express the transmitted signal as

$$m(t) = c(t)b(t) \tag{7.7}$$

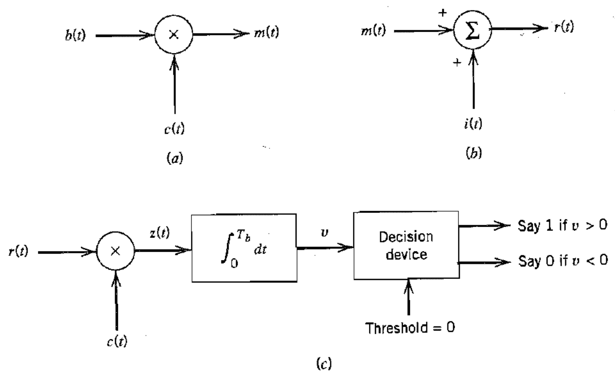

FIGURE 7.5 Idealized model of baseband spread-spectrum system. (*a*) Transmitter. (*b*) Channel. (*c*) Receiver.

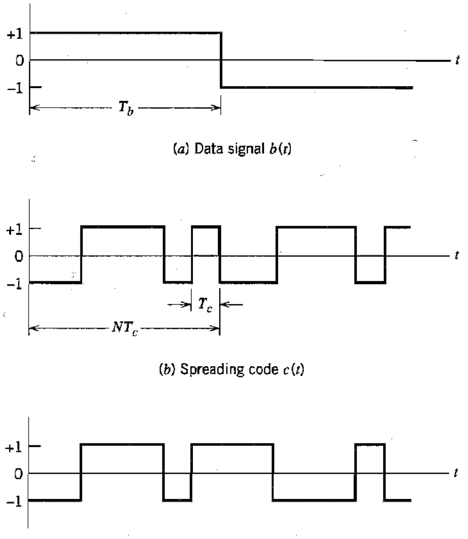

FIGURE 7.6 Illustrating the waveforms in the transmitter of Figure 7.5*a*.

The received signal $r(t)$ consists of the transmitted signal $m(t)$ plus an additive *interference* denoted by $i(t)$, as shown in the channel model of Figure 7.5*b*. Hence,

$$\begin{aligned} r(t) &= m(t) + i(t) \\ &= c(t)b(t) + i(t) \end{aligned} \tag{7.8}$$

To recover the original message signal $b(t)$, the received signal $r(t)$ is applied to a *demodulator* that consists of a multiplier followed by an integrator, and a decision device, as in Figure 7.5*c*. The multiplier is supplied with a locally generated PN sequence that is an exact *replica* of that used in the transmitter. Moreover, we assume that the receiver operates in perfect *synchronism* with the transmitter, which means that the PN sequence in the receiver is lined up exactly with that in the transmitter. The multiplier output in the receiver is therefore given by

$$\begin{aligned} z(t) &= c(t)r(t) \\ &= c^2(t)b(t) + c(t)i(t) \end{aligned} \tag{7.9}$$

Equation (7.9) shows that the data signal $b(t)$ is multiplied *twice* by the PN signal $c(t)$, whereas the unwanted signal $i(t)$ is multiplied only *once*. The PN signal $c(t)$ alternates between the levels -1 and $+1$, and the alternation is destroyed when it is squared; hence,

$$c^2(t) = 1 \qquad \text{for all } t \tag{7.10}$$

Accordingly, we may simplify Equation (7.9) as

$$z(t) = b(t) + c(t)i(t) \tag{7.11}$$