Spread Spectrum Encoding for Digital Signals

Richard W. DeVaul
MIT Media Laboratory
E'beth Haley
Harvard Physics

May 22, 1998
Revision: 1.1

Abstract

This paper documents a project in which we implemented a LFSR-based digital spread spectrum encoding and decoding system. The mathematics of spread-spectrum encoding are briefly discussed and a hardware implementation of spread spectrum encoding and decoding are presented. The steps necessary to go from spread-spectrum encoding to a full wireless communications system employing strong encryption are briefly described.

Contents

1 Introduction

Spread spectrum encoding is a technique for encoding data which makes maximal use of channel bandwidth, which allows for multiple signals to use the same channel without collision, and which is highly resistant to both interference and jamming. When combined with strong encryption and wireless transmission, this technology can be use to make secure, robust, wireless networks. This paper focuses on an elegant technique for spread spectrum encoding and decoding which requires a minimal amount of hardware.

2 Digital Spread Spectrum Encoding

In digital spread spectrum encoding, a stream of data bits is combined with deterministic noise to produce stream of encoded bits. These encoded bits have the desirable feature of appearing very noisy — up to the repeat length of the noise source there is no apparent structure in the encoded data stream. This is desirable, because this stream of bits can then be transmitted in such a way as to completely utilize the bandwidth of the transmission channel.

For instance, assume you want to send eight bit values, and have 256 discrete frequencies in a channel. A simple transmission scheme would be to represent each byte as a pulse on its corresponding frequency[1]. If you were sending ASCII text repressing this document, you would find that some frequencies were used much more than others (in fact, you would be using only one half of your channel since ASCII values are always between zero and 127). This would be a poor use of your spectral real-estate.

If on the other hand you were to spread your data using an LFSR, you would find that all of your frequencies were being used equally. Thus, you would be making maximal use of the channel, and the power spectrum across the discrete frequencies would be flat.

3 LFSR Signal Spreading

Spread spectrum encoding requires deterministic noise, or pseudo-random numbers. One way of generating these numbers is with a linear-feedback shift register. The noise produced by LFSRs has excellent spectral properties which make it ideal for spreading a signal, and the method for producing this noise is simple and easily implemented in digital logic.

[1] This probably isn't a very good modulation scheme, but it does illustrate the advantages of signal spreading in channel usage.

3.1 The Mathematics of LFSR Digital Signal Spreading

Note: The algorithm described here comes from the Spring 1998 notes for Neil Gershenfeld's "Physics of Information Technology" class, which are soon to be published in book form.

Given a shift register of length n, a tap vector $\mathbf{C} \in (G(2))^n$ (a vector of bits of length n), and a data bit $\mathbf{d}_i^{tx} \in G(2)$, the next encoded bit $\mathbf{x}_i^{tx}$ is calculated by

$$\mathbf{x}_i^{tx} = \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{tx} \cdot \mathbf{C}_j \right) + \mathbf{d}_i^{tx}, \tag{1}$$

where addition is define as logical XOR and multiplication as logical AND. Given a shift register of the same length, an identical tap vector, and a received bit $\mathbf{x}_i^{rx}$, a data bit $\mathbf{d}_i^{rx}$ is decoded from the encoded stream as

$$\mathbf{d}_i^{rx} = \mathbf{x}_i^{rx} + \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{rx} \cdot \mathbf{C}_j \right). \tag{2}$$

Assuming $\mathbf{x}_i^{tx} = \mathbf{x}_i^{rx}$, then

$$\mathbf{d}_i^{rx} = \mathbf{x}_i^{tx} + \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{tx} \cdot \mathbf{C}_j \right), \tag{3}$$

$$\mathbf{d}_i^{rx} = \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{tx} \cdot \mathbf{C}_j \right) + \mathbf{d}_i^{tx} + \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{tx} \cdot \mathbf{C}_j \right), \tag{4}$$

but since addition and subtraction are equivalent in this basis,

$$\mathbf{d}_i^{rx} = \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{tx} \cdot \mathbf{C}_j \right) - \sum_{j=1}^{n} \left(\mathbf{x}_{i-j}^{tx} \cdot \mathbf{C}_j \right) + \mathbf{d}_i^{tx}, \tag{5}$$

$$\mathbf{d}_i^{rx} = \mathbf{d}_i^{tx}. \tag{6}$$

Thus, if transmission occurs without error, the original signal can be reconstructed perfectly. Even assuming noisy transmission, encoding schemes can be used on top of the LFSR spreading to ensure no data is lost.

4 Project Summary

The most time-consuming part of the project was settling on what we wanted to do! Once we decided to work on a spread-spectrum system, the rest came relatively quickly.

4.1 Design phase

Once we narrowed the problem down to building the digital encoding and decoding system, it did not take long to design. Most of the time (a few days) was spent on research. Once we found the algorithm detailed above, it only took a few hours to design the basic circuit.

4.1.1 Our LFSR Hardware Design

It is surprisingly simple to translate Equation 1 and 2 into digital hardware using shift registers and a few gates. Please see the attached schematics for the details of our implementation, including a proposed scheme to keep the clock of the receiving register in sync with the transmitter.

4.2 Implementation

Implementing our design was fast, because the circuits turned out to be very simple. The stages of the hardware implementation were as follows:

1. We got our transmitter and receiver working on a single breadboard, sharing clocks, power supplies, and transmission by wire. Our input data stream was the state of a pushbutton, and our output was a red LED. At this point we kept the clock rate fairly low.

2. We implemented the circuit on separate breadboards and capacitively coupled the transmission line to the receiver, thus allowing the use of separate power supplies. Both transmitter and receiver still shared clock signal. Clock rates above the f3db point of our coupling worked fine. We tested the circuits up to 500kHz.

3. We used separate clocks for both transmitter and receiver (two function generators plugged into different power strips and separated by several feet to prevent accidental entrainment) in order to see how well the system performed with incompletely synchronized clocks. At this point we were using a single long wire as our transmission line, which undoubtedly introduced some RF noise. Even with some clock drift, the system continued to work very well, and could be reliably synchronized by hand at 500kHz.

Stages we didn't get to:

1. Implementing a phase-locked loop or other auto-synchronizing mechanism for the LFSR clocks, although we did come up with a proposed schematic for doing this.

2. Sending more interesting data, such as digitized audio.

3. Microcontroler or microprocessor based strong encryption front end.

5 A Complete System

Originally, we were thinking about this project as a component in a complete system for secure wireless digital communication. A complete system based on LFSR spread spectrum and strong encryption would require the following additional elements:

1. A phase-locked loop or some other technique to make sure the transmitting and receiving shift registers stay in sync.

2. Error-correction encoding combined with some kind of low-level packet scheme.

3. A microprocessor or Microcontroler implementing a strong encryption algorithm for both sender and receiver.

4. A modulation scheme for transmitting the spread digital data in RF.

We made some headway on the first problem, and spent some time talking about the others. The hardest of these is probably designing the error-correcting encoding scheme in such a way that it could be implemented in hardware to support high data rates, and the RF design (if we were to do that part ourselves). The encryption front-and-back end is mostly a software problem, and the rest is digital interfacing.

6 Conclusions

Building the complete communication system described above would undoubtedly take months, but the portion of it we were able to tackle was an interesting project in its own right. This was one of the rare instances in which our design got progressively simpler as we understood the problem better, and eventually became so simple as to be almost disappointing! However, the very simplicity of the LFSR signal spreading scheme is its strength, as we could probably push our clock rates to 50MHz (according to the spec sheets) using the components on our schematic without difficulty.