

PN Sequences and Scramblers

After this lecture you should be able to: distinguish between random and pseudo-random signals, classify signals as PN, PRBS, and/or ML PRBS signals according to their quantization, periodicity, mean value and maximum run lengths, draw the schematic of a LFSR ML PRBS generator, explain two reasons why scrambling may be desirable, select between scrambling and encryption based on the need for secrecy, select between additive and multiplicative scramblers based on the availability of framing information, explain the error patterns resulting from erroneous input to a self-synchronizing scrambler, and implement (draw schematic of) additive scramblers and self-synchronizing multiplicative scramblers.

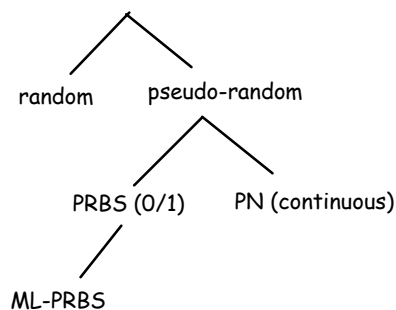
Random and Pseudo-Random Signals

Random signals are unpredictable. An example is the thermal noise generated by a resistor. Some statistics of the noise waveform, such as the average voltage (e.g. 0), the power (given by $kTBF$), and the spectrum (e.g. constant) may be known. But we may not be able to predict the value of the waveform at any given point in time.

It is sometimes useful to generate waveforms that both have known statistical properties (e.g. the average, power, spectrum, etc.) and whose values are predictable. These types of signals are called “pseudo-random” signals. Because of their noise-like characteristics they are also called pseudo-noise (PN) signals.

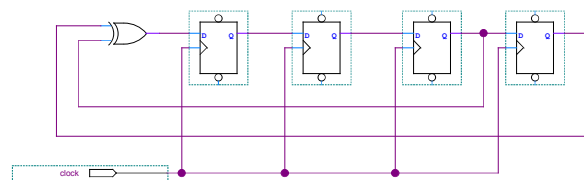
A pseudo-random bit sequence (PRBS) is a two-valued (0,1) PN signal. PN and PRBS signals have many important applications in communications systems. In this lecture we will study the properties of a type of PRBS called a maximal-length (ML) sequence, see how these sequences are generated and one of their applications, “scrambling.” Other applications include spread-spectrum systems and generating test signals.

So we have the following taxonomy of random signals:



Generating a ML PRBS

A ML PRBS can be implemented using a shift register whose input the modulo-2 sum of other taps.



This is known as a linear-feedback shift register (LFSR) generator. There are published tables showing the LFSR tap connections that result in a ML PRBS generator.

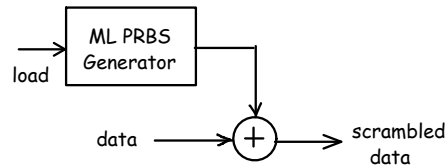
If the contents of the shift register ever become all zero then all future values will be zero. This is why the generator has only $2^K - 1$ states – the state corresponding to all zeros is not allowed.

Properties of a ML PRBS

ML PRBS sequences, sometimes called m-sequences, have a number of interesting properties including:

- the sequence is called maximum-length because the sequence has a period of $2^K - 1$ where K is the number of bits of state in the generator (shift register stages). This is one less than the maximum number of states of a K -bit counter.
- the sequence has exactly 2^{K-1} ones and $2^{K-1} - 1$ zeros
- there is 1 run of $K-1$ 1's, 2 runs of $K-2$ 1's and $K-2$ 0's, 3 runs of $K-3$ 1's, ...

- the generator runs through every possible set of states except all-zero,
- the sequence is approximately orthogonal to any shift of itself



Exercise 1: How many flip-flops would be required to generate a ML PRBS of period 16383? How many ones would the sequence have?

Scrambling

Much real-world data contains repetitive components. Examples include sequences of constant values such as a video image that doesn't change from one frame to the next, a document with sections that are all one level (e.g. white), or repetitive data values in a file or database.

Two possible problems are introduced by this non-random data:

- when these values are transmitted the periodic components of the signal result in peaks in the spectrum that have larger than average power. These strong discrete frequency components can cause interference to wireless devices.
- long sequences of certain values will result in a signal that may not have enough transitions to allow for clock recovery.

To solve these problems most communication systems use “scramblers” to remove periodicities or long constant sequences in the data. Two types of scramblers as described below.

It is important to understand that a scrambler does not provide secrecy (encryption).

Exercise 2: Why not?

Frame-Synchronous Scramblers

The simplest type of scrambler consists of a ML PRBS generator whose output is exclusive-ORed with the data. These types of scramblers are called “additive” scramblers because the PN sequence is added, modulo-2, to the data.

Since the ML sequence needs to be synchronized between the transmitter and receiver, this type of scrambler is only practical for systems that have a frame structure. The state of the ML PRBS generator can be set to a specific value at the start of each frame. This value can be either a fixed value for every frame or it can be a value that is included in the frame's preamble.

Self-Synchronizing Scramblers

Some protocols don't use framing and operate on a continuous sequence of bits. A scrambler for such a system needs to synchronize the descrambler to the scrambler without any external information in order to recover from any loss of synchronization.

Self-synchronizing scramblers are sometimes called multiplicative scramblers because scrambling and descrambling are implemented using polynomial division and multiplication. The scrambled output, $S(x)$, is generated at the transmitter by dividing by a generator polynomial $G(x)$:

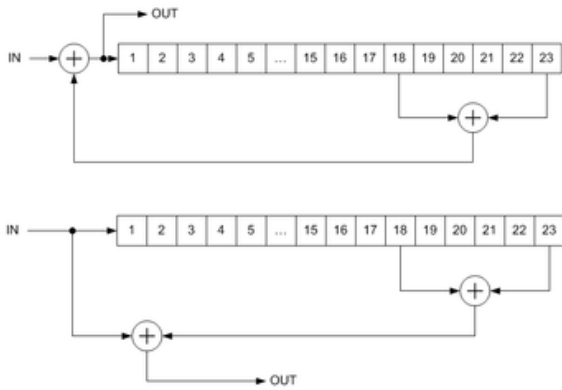
$$S(x) = \frac{M(x)}{G(x)}$$

and transmitting the quotient (the remainder is ignored). The receiver de-scrambles the scrambled signal by multiplying by $G(x)$:

$$M(x) = S(x)G(x)$$

As shown in a previous lecture we can implement polynomial division and multiplication using shift registers and xor gates.

For example, the ITU-T V.34 modem specification defines a self-synchronizing scrambler for calling mode that uses the generating polynomial: $GPC(x) = 1 + x^{-18} + x^{-23}$ (negative powers of x are used to indicate that higher-numbered orders correspond to earlier bits). The scrambler and descrambler can be implemented as shown in the following figures (numbers in boxes are delays, not bit index):



Exercise 4: What two signals would the receiver compare to detect errors?

One problem with self-synchronizing scramblers is that an error in the received data pattern can result in multiple errors in the de-scrambled errors. This is called error propagation.

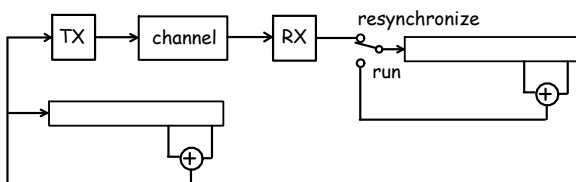
Exercise 3: How many errors will appear in the output of a V.34 descrambler if there is one input error?

Certain input sequences could set the scrambler state to zero and produce no scrambling. However the scrambler and descrambler can be designed to detect such undesirable sequences and invert the next bit when this is detected.

PRBS Test Sequences

We can test a communication system by transmitting a PRBS sequence over the channel and comparing the received sequence to a locally-generated copy. Since the hardware to generate a very long ML-PRBS is very simple, it is practical to use long sequences for testing.

One problem that arises is how to synchronize the transmitter and receiver. This can be done by loading the receiver PRBS generator's shift register with any K received bits. As long as there were no errors in these K bits then from that point on the transmit and receive generators will generate the same sequences:



If the receiver ever becomes un-synchronized with the transmitter the error rate will become very high. When this is detected at the receiver the local PRBS generator can resynchronize as above.