



Matlab, simulink

Building a Dircet sequence spread spectrum Model

Data Communications

CE00036-3

Tutors: Dr Alison Griffiths & Amr El-HELW

Email: a.l.griffiths@staffs.ac.uk elhelw@staffs.ac.uk

Building a Direct sequence spread spectrum Model

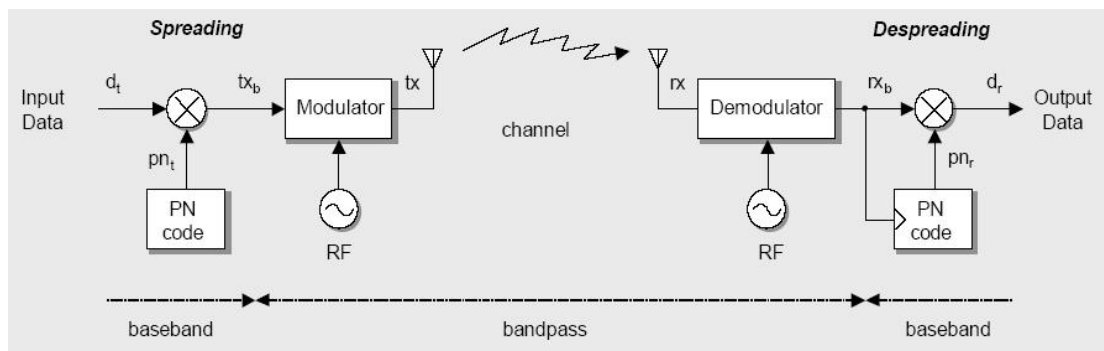
Introduction

A transmission technique in which a pseudo-noise code, independent of the information data, is employed as a modulation waveform to “spread” the signal energy over a bandwidth much greater than the signal information bandwidth. At the receiver the signal is “despread” using a synchronized replica of the pseudo- noise code

Direct Sequence Spread Spectrum (DS-SS)

Direct Sequence Spread Spectrum (DS-SS) is the most common version of spread spectrum in use today, due to its simplicity and ease of implementation. This method has been adopted by Qualcomm in designing their wireless communications network as well as by the Infopad project here at UC Berkeley.

. In DS-SS, the carrier (data signal) is modulated by the PN code sequence, which is of a much higher frequency than the desired data rate



What are PN Sequences?

A simple definition of PN Sequence

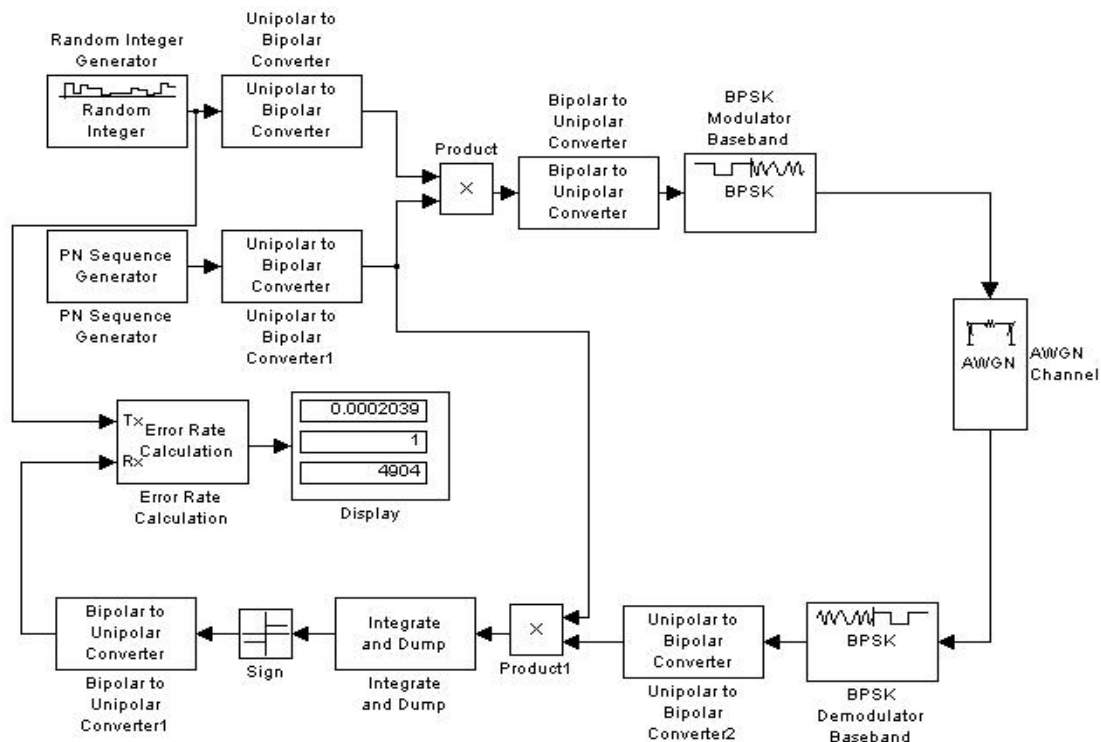
- Pseudo-random noise sequences or PN sequences are known sequences which exhibit the properties or characteristics of random sequences.
- PN Sequences can be used to logically isolate users on the same physical (frequency) channel. They can also be used to perform scrambling as well as spreading and de-spreading functions.
- If the Code sequence were deterministic, everybody could access the channel;
- If the Code sequence were truly random, then nobody, including the intended receiver, could access the channel;

So, PN sequences appear as random noise to everybody else, except to the transmitter and the intended receiver.

CDMA and DS-SS

Now, in order to facilitate a multi-user environment, all that needs to be done is to apply the principle of Code Division Multiple Access (CDMA). In a CDMA system, each user is identified by its own code, and in order to prevent users from interacting with each other, this code designed to be orthogonal to each other (the cross-correlation between any two of these codes is identically zero)

The following model simulates the use of direct sequence spread spectrum to send a signal through a channel with noise.



The topics in this section are as follows:

- [Building the DSSS Model](#)
- [Understanding the Blocks in the Model](#)
- [Setting Parameters in the DSSS Model](#)

Building the DSSS Model

To build the model, follow these steps:

1. Drag the following blocks from the Simulink Library Browser into the model window, and connect them as shown in the figure:
 - Random integer generator, from the data sources sublibrary of comm sources library (communication blockset).
 - Unipolar to bipolar converter, from the utility blocks sublibrary of the communication blockset library
 - PN sequence generator, from the sequence generators sublibrary of comm sources library (communication blockset).
 - Product, from the math operations sublibrary of the simulink library.
 - Bipolar to unipolar converter, from the utility blocks sublibrary of the communication blockset library
 - BPSK Modulator and Demodulator Baseband, from PM in the Digital Baseband Modulation sublibrary of the Modulation library (communication blockset).
 - AWGN Channel, from the channels (communication blockset).
 - Integrate and dump, from the comm filters sublibrary of the communication blockset library.

- Sign, from the math operations sublibrary of the simulink library.
- Error rate calculation from the comm sinks library (communication blockset).
- Display form the sources sublibrary of the simulink library.

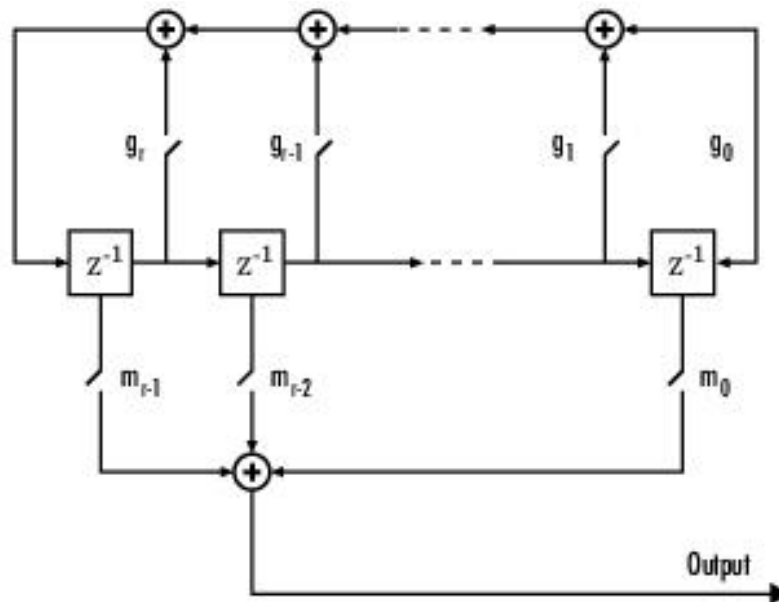
Understanding the Blocks in the Model

The model contains the following blocks.

The PN Sequence Generator

The PN Sequence Generator block generates a sequence of pseudorandom binary numbers. A pseudonoise sequence can be used in a pseudorandom scrambler and descrambler. It can also be used in a direct-sequence spread-spectrum system.

The PN Sequence Generator block uses a shift register to generate sequences, as shown below.



All r registers in the generator update their values at each time step according to the value of the incoming arrow to the shift register. The adders perform addition modulo 2. The shift register is described by the Generator Polynomial parameter, which is a primitive binary polynomial in z , $g_r z^r + g_{r-1} z^{r-1} + g_{r-2} z^{r-2} + \dots + g_0$. The coefficient g_k is 1 if there is a connection from the k^{th} register, as labeled in the preceding diagram, to the adder. The leading term g_r and the constant term g_0 of the Generator Polynomial parameter must be 1.

You can specify the Generator polynomial parameter using either of these formats:

- A vector that lists the coefficients of the polynomial in descending order of powers. The first and last entries must be 1. Note that the length of this vector is one more than the degree of the generator polynomial

- A vector containing the exponents of z for the nonzero terms of the polynomial in descending order of powers. The last entry must be 0.

For example, [1 0 0 0 0 1 0 1] and [8 2 0] represent the same polynomial, $p(z) = z^8 + z^2 + 1$.

The Initial states parameter is a vector specifying the initial values of the registers. The Initial states parameter must satisfy these criteria:

All elements of the Initial states vector must be binary numbers.

The length of the Initial states vector must equal the degree of the generator polynomial. Note At least one element of the Initial states vector must be nonzero in order for the block to generate a nonzero sequence. That is, the initial state of at least one of the registers must be nonzero. Different initial states lead to orthogonal code

Unipolar to Bipolar Converter

The Unipolar to Bipolar Converter block maps the unipolar input signal to a bipolar output signal. If the input consists of integers between 0 and $M-1$, where M is the M -ary number parameter, then the output consists of integers between $-(M-1)$ and $M-1$. If M is even, then the output is odd, and vice-versa

Integrate and Dump

The Integrate and Dump block creates a cumulative sum of the discrete-time input signal, while resetting the sum to zero according to a fixed schedule. When the simulation begins, the block discards the number of samples specified in the Offset parameter. After this initial period, the block sums the input signal along columns and resets the sum to zero every N input samples, where N is the Integration period parameter value. The reset occurs after the block produces its output at that time step.

The integrate-and-dump operation is often used in a receiver model when the system's transmitter uses a simple square-pulse model. It can also be used in fiber optics and in spread-spectrum communication systems such as CDMA (code division multiple access) applications.

Sign

The Sign block indicates the sign of the input: The output is 1 when the input is greater than zero. The output is 0 when the input is equal to zero. The output is -1 when the input is less than zero.

Setting Parameters in the DSSS Model

To set parameters in the convolutional code model, do the following:

1. Double-click the random integer generator block and make the following changes
 - Set **M-ary** number to 2.
 - Set **sample time** to 1/1000.
2. Double-click the PN sequence generator block and make the following changes to the default parameters in the block's dialog:
 - Set **sample time** to 1/32000 .

3. Double-click the Unipolar to bipolar converter block and make the following changes to the default parameters in the block's dialog:
 - Set **M-ary** number to 2.

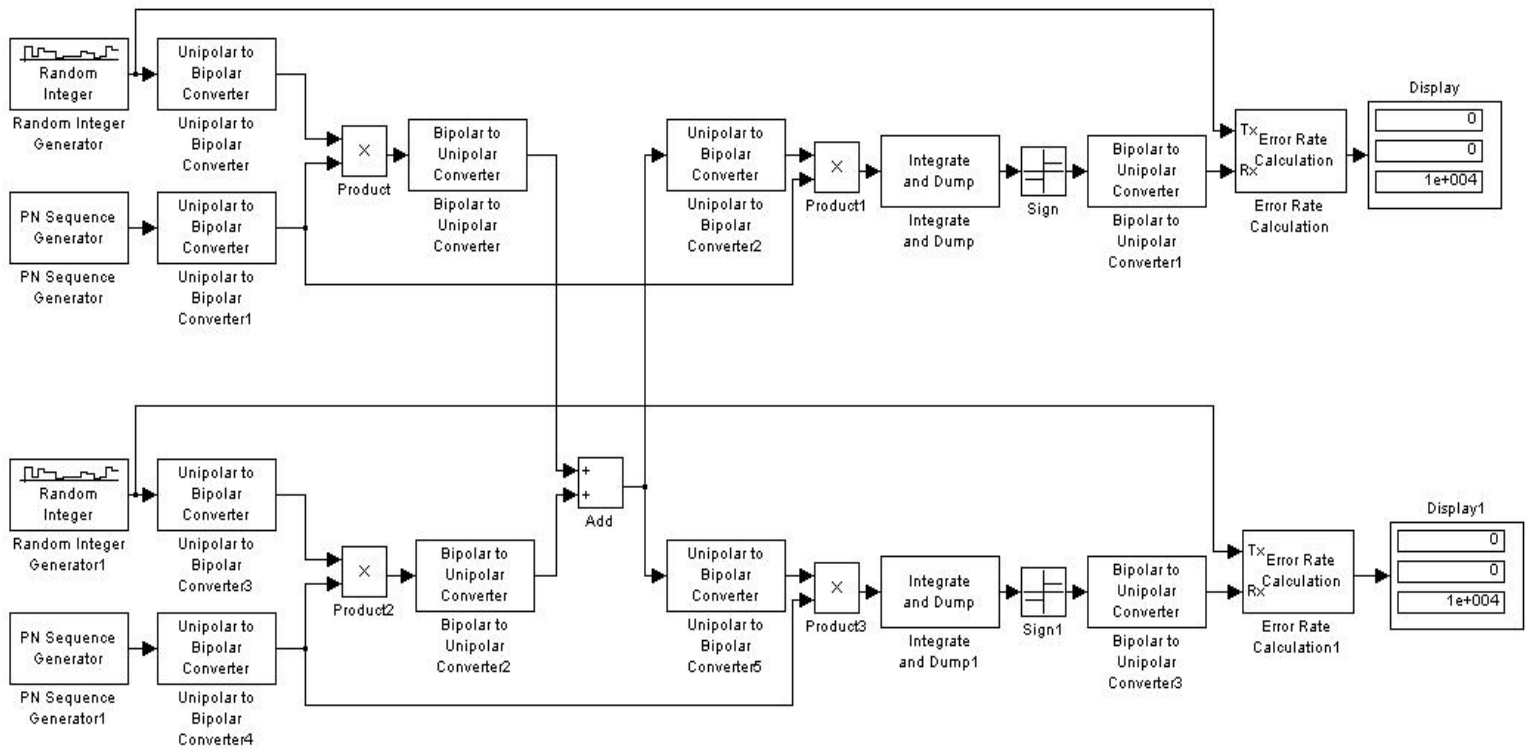
4. Double-click the bipolar to unipolar converter and make the following changes to the default parameters in the block's dialog:
 - Set **M-ary** number to 2.

5. Double-click the AWGN Channel block and make the following changes to the default parameters in the block's dialog:
 - Set **Es/No** to 6.
 - Set **Symbol period** to $1/32000$.

6. Double-click the Integrate and Dump block and make the following changes to the default parameters in the block's dialog:
 - Set **integration period** to 32.

7. Double-click the error rate calculation block and make the following changes
 - Set **receive delay** number to 1.

Now try to construct this model



note that: - The random interger generators must have different intial seeds to ensure two different independent sequences (users).

- assign to each user different PN code by usind different intial states to each one of them. For example [1 0 0 1 0 0] for the frist user and [1 1 0 0 0 0] for the second one.