1996

SIMULINK Implementation of a CDMA Transmitter

Visalakshi Ramakonar
Edith Cowan University

Recommended Citation

This Thesis is posted at Research Online.
https://ro.ecu.edu.au/theses_hons/311
Edith Cowan University

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study.

The University does not authorize you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following:

- Copyright owners are entitled to take legal action against persons who infringe their copyright.

- A reproduction of material that is protected by copyright may be a copyright infringement. Where the reproduction of such material is done without attribution of authorship, with false attribution of authorship or the authorship is treated in a derogatory manner, this may be a breach of the author’s moral rights contained in Part IX of the Copyright Act 1968 (Cth).

- Courts have the power to impose a wide range of civil and criminal sanctions for infringement of copyright, infringement of moral rights and other offences under the Copyright Act 1968 (Cth). Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.
SIMULINK Implementation of a CDMA Transmitter

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of Bachelor of Engineering (Communication Systems).

Visalakshi Ramakonar

Principal Supervisor: Dr Tadeusz Wysocki
Co-supervisor: Dr Hans Juergen Zepernick

November 1996

Faculty of Science, Technology, and Engineering
Department of Computer and Communication Engineering
Edith Cowan University
Western Australia
Abstract
An implementation of a Code Division Multiple Access (CDMA) transmitter has been developed using SIMULINK and MATLAB. This transmitter uses a modified carrier in modulation. This modified carrier, which is frequency modulated, has been shown to reduce intersymbol interference (ISI) and multiple access interference (MAI). These two types of interference are caused by multipath propagation which results in delayed versions of the original signal. The benefits of this modified modulation technique are apparent when there are delays involved. The spreading sequences used are 7-bit Gold codes which allow a maximum of nine users. The initial trials of the transmitter indicate that it is functioning correctly.
Acknowledgments

I would like to extend my appreciation and thanks to Dr Tadeusz Wysocki for his supervision and guidance throughout this project. I would also like to thank my parents and friends for their love and support.
I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any institution of higher education; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Signature

Date 

...........................
# Table of Contents

1 Introduction .......................................................................................................................... 1
   1.1 Motivation and Contributions of the Thesis ................................................................. 1
   1.2 Outline of the Thesis ....................................................................................................... 2

2 Spread Spectrum Communication and CDMA ................................................................. 3
   2.1 Introduction to Spread Spectrum (SS) Communication ............................................. 3
   2.2 Direct Sequence (DS) Spread Spectrum (SS) Signals ............................................. 4
      2.2.1 Processing Gain And Jamming Margin ............................................................... 6
   2.3 Frequency Hopping (FH) Spread Spectrum (SS) Signals .......................................... 7
   2.4 Synchronisation ........................................................................................................... 8
      2.4.1 Acquisition ........................................................................................................... 9
      2.4.2 Tracking .............................................................................................................. 11
   2.5 Code Division Multiple Access (CDMA) ................................................................... 12
      2.5.1 Transmitter Model ............................................................................................... 13
      2.5.2 Receiver Model ................................................................................................. 14
      2.5.3 Orthogonal Functions ......................................................................................... 14
      2.5.4 PN Code Sequence Generation ........................................................................... 15
      2.5.5 Gold and Kasami Sequences .............................................................................. 16
   2.6 Binary Phase Shift Keying (BPSK) Modulation ......................................................... 18
   2.7 Quadrature Phase Shift Keying (QPSK) Modulation ............................................... 19
   2.8 Benefits of CDMA over TDMA and FDMA ............................................................... 20
      2.8.1 Frequency Division Multiple Access (FDMA) .................................................. 20
      2.8.2 Time Division Multiple Access (TDMA) ........................................................... 21
2.8.3 Advantages of DS-CDMA 22

2.9 Near-Far Problem 24

2.9.1 Power Control 25

2.9.2 Solution 25

2.10 Handoff 25

2.11 Multiuser Detection 26

2.12 Multipath Channels 27

2.12.1 Fading Multipath Channels 28

2.13 Intersymbol Interference (ISI) and Multiple Access Interference (MAI) 30

2.13.1 Intersymbol Interference (ISI) 30

2.13.2 Multiple Access Interference (MAI) 31

2.14 Reduction Of ISI and MAI 31

2.14.1 Multipath in DS-CDMA 32

2.14.2 Carrier Waveform Modification 32

3 SIMULINK Implementation of a CDMA Transmitter 34

3.1 Proposed Transmitter Model 34

3.2 SIMULINK Overview 34

3.2.1 Perceptiveness 35

3.2.2 Convenience 35

3.2.3 Flexibility 36

3.2.4 Modularity 36

3.2.5 Power of procedural language 36

3.2.6 S-functions 36

3.2.7 Combining MATLAB Functions With SIMULINK 37
3.3 SIMULINK Implementation of a CDMA Transmitter Using BPSK

3.4 SIMULINK Implementation of a CDMA Transmitter Using QPSK

4 Simulation Trials

4.1 Results Using BPSK Modulation

4.2 Results Using QPSK Modulation

4.3 Cross Correlation Values

4.4 Cross Correlation Values With Delay

4.4.1 The Cross Correlation Coefficients Between Signals Using BPSK Modulation With the W(t) Function.

4.4.2 The Cross Correlation Coefficients Between Signals Using BPSK Modulation Without the W(t) Function.

4.4.3 Analysis

4.5 Average Power Spectral Density (PSD) of the Output Signals

4.5.1 Average PSD of the Output Signals Using BPSK Modulation

4.5.2 Average PSD of the Output Signals Using QPSK Modulation

5 Further Development: Digital Signal Processing (DSP) Implementation

5.1 Benefits of Using DSP

5.2 Hardware Description

6 Conclusions

References

Appendix A: SIMULINK Implementation of a CDMA Transmitter Using BPSK (sim5g.m)

Appendix B: SIMULINK Implementation of a CDMA Transmitter Using QPSK (simq13.m)
Acronyms

**BER**  Bit Error Rate

**BPSK**  Binary Phase Shift Keying

**CDMA**  Code Division Multiple Access

**DS**  Direct Sequence

**DSP**  Digital Signal Processing

**FH**  Frequency Hopping

**FDMA**  Frequency Division Multiple Access

**ISI**  Intersymbol Interference

**MAI**  Multiple Access Interference

**TDMA**  Time Division Multiple Access

**QPSK**  Quadraphase Shift Keying

**SS**  Spread Spectrum
1 Introduction

1.1 Motivation and Contributions of the Thesis

One of the areas of telecommunications that has attracted much research interest in the last few years is indoor wireless communications, particularly wireless local area networks (WLANs). This has been due to the growth of computer based applications in almost all work environments [12]. Current WLANs are designed for high speed data transmission and will mainly operate or are currently operating on industrial scientific medical (ISM) bands [3]. At these bands, spread spectrum (SS) methods in the form of either direct sequence spread spectrum (DS SS) or frequency hopping spread spectrum (FH SS) can be used in order to improve system performance. There is, however, a significant bit error rate (BER) degradation when the data rate is of several Mbps, which is typical of asynchronous transfer mode (ATM) WLANs. This BER degradation is caused by the channel dispersions due to multipath propagation [3]. Multipath effects cause serious distortions to the received signal. To combat multipath effects, Code Division Multiple Access (CDMA) and DS techniques are implemented in the advanced levels [12].

The aim of this project was:

To develop a SIMULINK implementation of a CDMA transmitter.

This thesis presents an original work in the design and implementation of a CDMA transmitter. The transmitter is capable of generating a modulated data sequence multiplied by a spreading sequence called a PN sequence. It uses a non-conventional method of modulation using a function called the $W(t)$ function. This function has been shown to reduce intersymbol interference (ISI) and multiple access interference (MAI) which are both caused by multipath propagation [3].

The transmitter was implemented in SIMULINK and MATLAB. The MATLAB function blocks in SIMULINK were used to implement simple functions needed
in the development of the transmitter. The predefined SIMULINK blocks given in the SIMULINK menu proved sufficient in the construction of the transmitter.

1.2 Outline of the Thesis

The outline of the thesis is as follows:

- Chapter 2 provides the theoretical foundation required to understand spread spectrum techniques and CDMA. It also outlines a method of reducing multiple access interference (MAI) and intersymbol interference (ISI) which are both caused by multipath. The near-far effect (which is common with CDMA) and power control, used to reduce this effect, are also both discussed.

- Chapter 3 describes the SIMULINK implementation of the transmitter. It also introduces SIMULINK and its usefulness in modelling communication systems.

- Chapter 4 examines the results from the simulations where the transmitter was tested. The cross correlation coefficients and the power spectral density of the output signals are assessed.

- Chapter 5 outlines further development of the SIMULINK model to implement a hardware version of the transmitter using digital signal processing (DSP) techniques. This chapter also outlines the benefits of using DSP as well as a description of the hardware specifications of the DSP board that would be suitable for this project.

- Finally, chapter 6 concludes the thesis by summarising the major outcomes of the project.
2 Spread Spectrum Communication and CDMA

2.1 Introduction to Spread Spectrum Communication

Code Division Multiple Access (CDMA) is a form of spread spectrum multiple access communication. Spread spectrum signals have the characteristic that their bandwidth $W$ is much greater than the information rate $R$ in bits/s. The bandwidth expansion factor $B_e$ can be described as:

$$B_e = \frac{W}{R} \quad (1)$$

There are high levels of interference that are present in the digital transmission of information over some radio channels. This interference can be overcome by using a larger bandwidth than the minimum required.

Spread spectrum signal design also incorporates an element of pseudo-randomness which makes the signals appear similar to random noise. This makes the message difficult to be intercepted and only allows the intended receivers to demodulate the signal.

Thus apart from the use of SS for multiple access, spread spectrum signals are used for [1]:

- "combatting or suppressing the detrimental effects of interference due to jamming, interference arising from other users of the channel, and self interference due to multipath propagation;
- hiding a signal by transmitting it at a lower power and, thus, making it difficult for an unintended listener to detect in the presence of background noise;
- achieving message privacy in the presence of other listeners."

When trying to prevent jamming, the communicators must not allow the jammer to have any previous knowledge of the signal characteristics except for the type of modulation and the overall channel bandwidth. If the information is just encoded, the jammer may be able to copy the transmitted signal and
SIMULINK Implementation of a CDMA Transmitter

confuse the receiver. To prevent this, the transmitter adds an element of pseudo-randomness to the signal that is only known by the receiver. In multiple-access communication systems, a number of users share a common channel bandwidth. Since any of the users may transmit information simultaneously over the channel to the corresponding receivers, interference may arise. If the users all use the same code for the encoding and decoding of the information, then each signal being transmitted may be differentiated by superimposing a different pseudo-random code, known as a PN sequence, onto each one. This pseudo-random sequence is also called a key. Hence a receiver may recover the transmitted information by knowing the key, thereby achieving message privacy. This is the communication technique used by CDMA [1].

There are two types of spread spectrum signals. These two types are Direct Sequence (DS) spread spectrum and Frequency Hopping (FH) spread spectrum signals. DS SS signals use phase shift keying (PSK) to modulate the binary data. FH SS signals use frequency shift keying (FSK) for modulation. DS SS is the technique used in this project however an explanation of both methods is given in sections 2.2 and 2.3.

2.2 Direct Sequence (DS) Spread Spectrum (SS) Signals

The type of modulation that will be considered here is phase modulation or phase shift keying. The phase of the PSK signal is shifted pseudo-randomly by the combination of the PN sequence generated at the modulator and the PSK modulation. This is true only for binary PSK (BPSK). The modulated signal is called a direct sequence (DS) or pseudo-noise (PN) spread spectrum signal.

Assume for a particular system that the information rate at the encoder is \( R \) bits/s, the available channel bandwidth is \( W \) Hz and BPSK is used. Then the phase of the carrier is shifted by \( \pi \) or \( 0 \) radians pseudo-randomly at a rate of \( W \) times/s according to the PN generator pattern. If quadruphase PSK (QPSK) is used then an amplitude modulation instead of a phase shift takes place. This allows the available channel bandwidth to be utilised.
The duration of a rectangular pulse called a chip, can be defined as

\[ T_c = \frac{1}{W} \]  

(2)

where \( T_c \) is called the chip interval.

The duration of a rectangular pulse that corresponds to a bit of information can be defined as

\[ T_b = \frac{I}{R}. \]  

(3)

The bandwidth expansion factor can then be expressed as

\[ B_e = \frac{W}{R} = \frac{T_b}{T_c}. \]  

(4)

The ratio \( T_b/T_c \) is usually an integer described as

\[ L_c = \frac{T_b}{T_c}. \]

This is the number of chips per information bit that occur in the transmitted signal during the bit duration.

If the encoder in a DS spread spectrum system takes \( k \) information bits at time and uses a binary linear \((n,k)\) block code, the time duration for transmitting the \( n \) code elements is \( kT_b \) seconds. The number of chips that occur in this time interval is \( kL_c \). Thus the block length of the code is \( n_c = kL_c \). If the encoder uses convolution coding at a rate of \( k/n \), then the number of chips in the interval \( kT_b \) is also \( n_c = kL_c \).

A method for superimposing the PN sequence onto the transmitted signal for BPSK, is to add the PN sequence to the coded bits with modulo-2 addition. If \( c_i \) represents the \( i \)-th bit of the PN sequence and \( b_i \) is the corresponding bit from the encoder, then the modulo sum is \([1]\):

\[ a_i = b_i \oplus c_i \]

The sequence \( \{a_i\} \) is then mapped into a BPSK signal of the form \([1]\)

\[ s(t) = \pm Re [g(t)e^{j2\pi f_0 t}] \]

where

\[ g(t) = \begin{cases} g(t - iT_c) & (a_i = 0) \\ -g(t - iT_c) & (a_i = 1) \end{cases} \]

(5)

and \( g(t) \) is a rectangular pulse of duration \( T_c \) and arbitrary shape. This is effectively an amplitude modulation rather than a phase shift.
The modulo-2 addition can also be represented as the multiplication of two waveforms [1].

The elements in the coded sequence can be mapped into a BPSK signal as in the relation

\[ b_i(t) = (2b_i - 1)g(t - iT_c). \]  \hspace{1cm} (6)

We can also define a waveform \( p_i(t) \) such that

\[ p_i(t) = (2c_i - 1) p(t - iT_c) \]  \hspace{1cm} (7)

\( p_i(t) \) is a waveform of duration \( T_c \).

The equivalent lowpass signal in relation to the \( i \)-th coded bit is [1]:

\[ g_i(t) = p_i(t)c_i(t) = (2b_i - 1) (2c_i - 1)g(t - iT_c) \]  \hspace{1cm} (8)

Hence it is shown that multiplying a BPSK signal that has been generated from the coded bits with a sequence of unit amplitude rectangular pulses (of duration \( T_c \)) and with a polarity that is determined from the PN sequence as in (7), is equivalent to the modulo-2 addition of the coded bits with the PN sequence followed by a mapping to give a BPSK signal.

### 2.2.1 Processing Gain and Jamming Margin

The performance characteristics of the DS spread spectrum signal can be found by expressing the signal energy per bit \( \xi_b \) in terms of the average signal power \( P_{av} \):

\[ \xi_b = P_{av}T_b \]

where \( T_b \) is the bit interval.

If \( J_o \) is the power spectral density of the jamming signal (wideband interference), then the total average jamming power is defined as

\[ J_{av} = J_o W \]

The ratio

\[ J_{av} / P_{av} \]

is known as the jamming margin. It is the largest value that the ratio can take while still achieving a given error rate performance.

The processing gain can be defined as in equation (4)
This ratio represents the advantage gained over the jammer that is achieved by expanding the bandwidth of the transmitted signal.

### 2.3 Frequency Hopping Spread Spectrum Signals

For another spread spectrum technique known as frequency hopping (FH), \( M \)-ary frequency shift keying (MFSK) is the most common modulation technique used. In this type of modulation, one of the \( M \) frequencies is used to determine which \( k = \log_2 M \) information bits are to be transmitted. The position of the \( M \)-ary signal is shifted pseudorandomly by the frequency synthesiser over a hopping bandwidth \( W_h \) [2]. In a conventional MFSK system, the data symbol modulates a fixed frequency carrier. In a FH/MFSK system, the data symbol modulates a carrier whose frequency is pseudorandomly determined. In both cases, a single tone is transmitted. A diagram of a FH/MFSK system is shown in Figure 2.1 [2].

![Figure 2.1 FH/MFSK System](image)

The diagram illustrates a two-step modulation process where the two steps are data modulation and frequency hopping modulation. The FH/MFSK system can also be implemented as a single step where the PN sequence and the data both determine a transmission tone produced by the frequency synthesiser. A PN generator feeds the frequency synthesiser a frequency word (a sequence of
Y chips) at each frequency hop time. This frequency word determines one of \(2^R\) symbol set positions. The minimum number of chips necessary in the frequency word is determined by the frequency hopping bandwidth \(W_s\) and the minimum frequency spacing \(\Delta f\) between consecutive hop positions.

The occupied transmission bandwidth for a given hop, is identical to the bandwidth of a conventional MFSK system, which is usually much smaller than \(W_s\). However, the FH/MFSK spectrum occupies the entire spread spectrum bandwidth when averaged over many hops. FH bandwidths can be in the order of several gigahertz [2]. This is much larger than can usually be achieved with DS SS and as a consequence, FH systems may have larger processing gains than DS systems. However, it is difficult to maintain phase coherence from hop to hop because of the wide bandwidth. Therefore these schemes are usually implemented using noncoherent demodulation.

As the diagram illustrates, the receiver reverses the transmitter's signal processing steps. The received signal is dehopped (FH demodulated) by mixing it with the same sequence of pseudorandomly selected frequency tones that was used for hopping. The most likely symbol is selected by sending the demodulated signal through a conventional bank of M noncoherent energy detectors.

2.4 Synchronisation

A receiver must use a synchronised copy of the spreading or code signal for the successful demodulation of the received signal. This is applicable for both DS and FH SS systems. There are two steps that are involved in the synchronisation of the locally produced spreading signal and the received SS signal. The first step is called acquisition and this consists of bringing the two spreading signals into rough alignment with one another. After acquisition of the SS signal, the second step called tracking is implemented. In this process, the best possible waveform fine alignment is continually maintained using a feedback loop.
2.4.1 Acquisition

The problem in acquisition is searching through a domain of time and frequency uncertainty so that the locally generated spreading signal is synchronised with the received SS signal. There are two types of acquisition methods that can be described as coherent or noncoherent. Most acquisition methods use noncoherent detection. This is because the despreading process is usually carried out before carrier synchronisation and thus the carrier phase is not known at this point. The following points must be taken into consideration when determining the limits of uncertainty in time and frequency [2]:

- Uncertainty in the distance between the transmitter and receiver. This determines the uncertainty in the amount of propagation delay.
- Phase differences between the transmitter and receiver spreading signals arise as a result of the clock instabilities between the transmitter and receiver. This phase difference grows as a function of elapsed time between synchronisation.
- The uncertainty in the value of the Doppler frequency offset of the incoming signal is a result of the uncertainty in the receiver's relative velocity with respect to the transmitter.
- Frequency offsets between the two signals are a result of the relative oscillator instabilities between the transmitter and the receiver.

2.4.1.1 Structures of the Correlator

In an acquisition method, the received signal and the locally generated signal are usually correlated first to produce a standard of similarity between them. It is then decided if the two signals are synchronised by comparing the correlated value to a threshold. If the two signals are not synchronised, the acquisition method implements a phase or frequency change in the locally generated code and another correlation is attempted. This is part of a systematic search through the receiver's phase and frequency uncertainty region. A DS parallel-search acquisition scheme will now be considered. In this system, the locally generated code is available with delays that are spaced a half chip (Tc/2) apart. 2Nc correlators are used if the time uncertainty between the local code and
the received code is $N_c$ chips and a complete parallel search of the whole time uncertainty area is to be completed in a single search time. A sequence of $\lambda$ chips are examined simultaneously by each correlator. After this, the $2N_c$ correlator outputs are compared. The locally generated code is chosen from the corresponding correlator output with the largest value. This is the simplest of the search techniques. It uses a maximum likelihood algorithm to find the code.

2.4.1.2 Serial Search

A common method for the acquisition of SS signals is to use a single correlator or matched filter to serially search for the correct phase of the DS code signal. The serial implementation repeats the correlation procedure for each possible sequence shift and as a result, reduction in complexity, size and cost can be achieved. In a DS scheme, the timing period of the local PN code is fixed and the locally generated PN sequence is correlated with the incoming PN sequence. The output signal is compared to a preset threshold at fixed examination intervals of $\lambda T_c$ (search dwell time), where $\lambda \gg 1$. The phase of the locally generated code signal is incremented by a fraction (typically one half) of a chip if the output is below the threshold. The correlation is then attempted again. When the threshold is surpassed, the PN sequence is assumed to have been obtained. This then prompts the code tracking procedure to be initiated and the phase-incrementing process of the local code is repressed.

2.4.1.3 Sequential Estimation

Another search technique is called Rapid Acquisition by Sequential Estimation (RASE) [2]. The RASE system inputs its best estimate of the first $n$ received PN code chips into the $n$ stages of its local PN generator. A starting date is defined by the fully loaded register. This date is from when the generator begins its operation. If the first $n$ received chips are correctly estimated, all the succeeding chips from the local PN generator will be correctly produced. This is because a PN sequence has the property that the next combination of register states depends only on the present combination of states. The RASE system has a switch that was initially at position one. The switch is now
thrown to position two. The local generator produces the same sequence as the incoming waveform, in the non-appearance of noise, if the starting state has been correctly estimated. We assume that synchronisation has occurred if the correlator output $\lambda T_e$ surpasses a pre-determined threshold level. If the output is less than the threshold, the switch is restored to position one and the procedure is repeated once the register is reloaded with estimates of the next $n$ received chips. The system no longer needs estimates of the input code chips once synchronisation has occurred.

The RASE system has a fast acquisition capability but it is subject to noise and interference signals.

2.4.2 Tracking

Tracking occurs once acquisition or rough synchronisation is achieved. There are two classifications of tracking code loops: coherent or noncoherent. Coherent loop is one in which the carrier phase and frequency are known exactly so that the loop can operate on a baseband signal. The carrier frequency and phase is not known exactly in a noncoherent loop. A noncoherent loop is usually used to track the received PN code because the carrier frequency and phase are not exactly known initially. Tracking loops can be categorised further as delay-locked loop (DLL) or as a tau-dither loop (TDL) [2].
2.5 Code Division Multiple Access

Code Division Multiple Access (CDMA) systems allow for many DS spread spectrum signals to share the same channel bandwidth provided that each signal has its own signature sequence (distinct PN sequence). Hence several users can transmit messages at the same time over the same channel bandwidth. The CDMA system block diagram is shown in Figure 2.2 [13].

![CDMA System Block Diagram](image)

**Figure 2.2 CDMA System Block Diagram**

The signature sequence is used to modulate and spread the signal containing the information. At the receiver, the signature sequence is also used to demodulate the message that had been transmitted by a user of the channel.

Suppose that a CDMA channel is shared by \( k \) simultaneous users and each user is assigned a signature waveform \( g_k(t) \) of duration \( T \). Where \( T \) is the symbol interval. The signature waveform can be expressed as

\[
g_k(t) = \sum_{n=0}^{L-1} c_k(n)p(t-nT_c), \quad 0 \leq t \leq T
\]  

(9)
where \( \{ c_k(n), 0 \leq n \leq L-1 \} \) is a pseudo-noise code sequence comprising \( L \) chips that can take the values \( \{ \pm 1 \} \) and \( p(t) \) is a pulse of duration \( T_c \). Therefore there are \( L \) chips per symbol since \( T = LT_c \).

Assume that all \( K \) signature waveforms have unit energy. Hence

\[
\int_0^T g_k^2(t) dt = 1. \tag{10}
\]

The cross correlations between pairs of signature waveforms are very important. We can define the cross correlations as [1]:

\[
p_{ij}(\tau) = \int_0^T g_i(t)g_j(t-\tau) dt, \quad i \neq j \tag{11}
\]

\[
p_\beta(\tau) = \int_0^T g_i(t)g_i(t+T-\tau) dt, \quad i \neq j \tag{12}
\]

2.5.1 Transmitter Model

Let the information sequence of the \( k \)th user be expressed as \( \{ b_k(m) \} \) where the value of each bit of information is \( \pm 1 \). If a block of data of length \( N \) is used, then the data block from the \( k \)th user is

\[
b_k = [b_k(1) \ldots b_k(N)]^T. \tag{13}
\]

Thus, the equivalent low-pass transmitted waveform is

\[
s_k(t) = \sqrt{\xi_k} \sum_{i=1}^{N} b_k(i) g_k(t-iT) . \tag{14}
\]

The composite, \( K \)-user's signal can therefore be described as

\[
s_k(t) = \sum_{k=1}^{K} \sqrt{\xi_k} \sum_{i=1}^{N} b_k(i) g_k(t-iT-\tau_k) \cos(\omega_c t + \theta_k), \tag{15}
\]

where \( \cos(\omega_c T + \theta_k) \) is the carrier waveform, \( \theta_k \) is the random carrier phase of the \( k \)th user, \( \xi_k \) is the energy signal per bit, \( \omega_c \) is the carrier frequency and \( \tau_k \) is the transmission delay. For synchronous CDMA, the delay \( \tau_k = 0 \) for all users. For asynchronous CDMA, the delays can be different. The duration of a data bit can be expressed as

\[
T = NT_c. \tag{16}
\]
2.5.2 Receiver Model

The signal at the receiver is [13]:

\[ r(t) = s(t) + n(t), \quad (17) \]

where \( n(t) \) is additive noise.

The receiver is made up of \( K \) parallel receivers corresponding to the \( K \) users. Each one multiplies the signal received by the carrier with an appropriate phase and the corresponding PN sequence. The received signal is given by [14]:

\[ r(t) = \sum_{k=1}^{K} \sqrt{\xi} \sum_{i=1}^{N} b_k(i) g_k(t - iT - \tau_k) \cos(\omega_c t + \phi_k) + n(t), \quad (18) \]

where \( \phi_k = (\theta_k - \omega_c \tau_k) \) and \( n(t) \) is the channel noise process which is assumed to be a white Gaussian process with two sided spectral density \( N_0/2 \).

When users are simultaneously transmitting messages, the PN code sequences used must be mutually orthogonal so that interference from other users is avoided. The orthogonality of the PN sequences is hard to achieve, especially if the number of sequences is large. So it is imperative that a good selection of PN sequences is obtained. This will be described in section 2.5.3.

2.5.3 Orthogonal Functions

Orthogonal functions are characterised by a set of \( N \) linearly independent functions \( \{\psi_j(t)\} \) called basis functions. The basis functions must satisfy the following conditions for the interval \( 0 \leq t \leq T \), on which they are said to be orthogonal:

\[ \int_0^T \psi_j(t) \psi_k(t) dt = K \delta_{jk} \quad k = 1, \ldots, N \quad (19) \]

where

\[ \delta_{jk} = \begin{cases} 1 & \text{for } j = k \\ 0 & \text{otherwise} \end{cases} \quad (20) \]
The operator \( \delta_k \) is called the *Kronecker delta function*. When the \( K_j \) constants are equal to one, then \( \Psi_j(t) \) is called an *orthonormal function*. The principle requirements for orthogonality are [2]:

- "Each \( \Psi_j(t) \) function of the set of basis functions must be linearly independent of the other members of the set.
- From a geometric point of view, each \( \Psi_j(t) \) is mutually perpendicular to each of the other \( \Psi_k(t) \) for \( j \neq k \)."

Hence, if a set is selected such that the sequences are orthogonal to one another, the transmitted signal when mixed with a PN sequence from the same set will also become orthogonal to any other signal being transmitted. Thus interference from the other users can be combatted.

2.5.4 PN Code Sequence Generation

The most widely known PN sequences are the maximum-length shift-register sequences (m-sequences) which have a length of \([1]\)

\[
n = 2^m - 1 \text{ bits.} \tag{21}
\]

They are generated by an m-stage shift register with linear feedback. The sequence is periodic with period \( n \) and each period of the sequence contains \( 2^m - 1 \) zeros and \( 2^m - 1 \) ones.

The binary sequence contains 1s and 0s and this sequence is mapped into a corresponding sequence of positive and negative polarity pulses. The relationship can be expressed as

\[
p(t) = (2b_1 - 1) p(t - iT). \tag{22}
\]

\( p(t) \) is the pulse corresponding to the element \( b_1 \) in the sequence which is either 1 or 0.

The periodic autocorrelation function defined in terms of the bipolar sequence is

\[
\phi(j) = \sum_{i=1}^{n} (2b_i - 1)(2b_{i+j} - 1), \quad 0 \leq j \leq n - 1 \tag{23}
\]

where \( n \) is the period.
A pseudo-random sequence should have an autocorrelation function with the property that \( \phi(0) = n \) and \( \phi(j) = 0 \) for \( 1 \leq j \leq n - 1 \). For \( m \) sequences, the periodic autocorrelation function is

\[
\phi(j) = \begin{cases} 
 n & (j = 0), \\
 -1 & (1 \leq j \leq n - 1).
\end{cases}
\]  

(24)

It is desirable in a CDMA system to have low cross-correlation values between a pair of sequences. However, the number of \( m \) sequences that have low cross-correlation values is too small for CDMA purposes. Therefore it has been found that the PN sequences with better periodic cross-correlation properties are Gold and Kasami sequences.

### 2.5.5 Gold and Kasami Sequences

Gold and Kasami found that certain pairs of \( m \) sequences of length \( n \) have a three-valued cross correlation function with the values \( \{-1, -t(m), t(m) - 2\} \), where

\[
t(m) = \begin{cases} 
 2^{(m+1)/2} + 1 & (\text{odd } m), \\
 2^{(m+2)/2} + 1 & (\text{even } m).
\end{cases}
\]  

(25)

Say, for example, that \( m = 5 \), then \( t(5) = 2^3 + 1 = 9 \). Then the three possible values of the periodic cross correlation function are \( \{-1, -9, 7\} \) and the maximum magnitude of the cross-correlation for the pair of \( m \)-sequences is 9.

Two \( m \)-sequences of length \( n \) with the values of the periodic cross-correlation taking on \( \{-1, -t(m), t(m) - 2\} \) are called preferred sequences. From a pair of preferred sequences of \( a = [a_1a_2...a_n] \) and \( b = [b_1b_2...b_n] \), a sequence of length \( n \) can be constructed by taking the modulo-2 sum of \( a \) with the \( n \) cyclically shifted versions of \( b \) or vice versa. The resulting new periodic sequences have period \( n = 2^m - 1 \). By including the original sequences \( a \) and \( b \) we have a total of \( n + 2 \) sequences called Gold Sequences. Refer to Figure 2.3 for a diagram of a Gold code set generator with period 63.

Kasami sequences have cross-correlation and autocorrelation values from the set \( \{-1, -(2^{m/2} + 1), 2^{m/2} - 1\} \). Thus the maximum cross-correlation value for any pair of sequences from the set is

\[
\phi_{\text{max}} = 2^{m/2} + 1
\]  

(26)
Kasami sequences are constructed by beginning with an $m$ sequence $a$, and forming a binary sequence $b$ by taking every $2^{m/2} + 1$ bit of $a$. This sequence, $b$, has period $n = 2^{m/2} - 1$. Then 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 $b$ and all $2^{m/2} - 2$ cyclic shifts of the bits from $b$. By including $a$ in the set, a set of $2^{m/2}$ Kasami sequences of length $n = 2^m - 1$ is obtained. Refer to Figure 2.4 for a diagram of a small Kasami set generator with period 63.

![Figure 2.3 Gold Code Set Generator of Period 63](image1)

![Figure 2.4 Small Kasami Set Generator of Period 63](image2)
2.6 Binary Phase Shift Keying Modulation

BPSK is one of the modulation techniques used in this project. In a phase shift keying system, a sinusoidal carrier wave of fixed amplitude and fixed frequency is used to represent the binary values 0 and 1. The modulating data signal shifts the phase of the waveform \( s_i(t) \) to one of two states, either zero or \( \pi \). The general expression for BPSK is:

\[
 s_i(t) = \sqrt{\frac{2E_i}{T_b}} \cos \left( \omega_c t + \phi_i(t) \right) \quad 0 \leq t \leq T_b \\
i = 1, 2
\]  

(27)

where \( E_i \) is the transmitted signal energy per bit and \( T_b \) is the symbol time duration.

To generate a BPSK wave, the input binary sequence must be represented in a bipolar form with symbols 1 and 0 representing constant amplitude levels of \( +\sqrt{E_i} \) and \( -\sqrt{E_i} \) respectively. This binary wave is multiplied by a sinusoidal carrier wave \( \phi_i(t) \) with frequency \( f_c = n_c T_b \) where \( n_c \) is a fixed integer and \( \phi_i(t) \) is defined as:

\[
 \phi_i(t) = \frac{2}{T_b} \cos(2\pi f_c t).
\]  

(28)

The transmitter model is shown in Figure 2.5.

---

Figure 2.5 BPSK Transmitter Model
2.7 Quadriphase Shift Keying Modulation

QPSK is the other modulation technique used in this project. In QPSK modulation, the original data stream, $d_k(t) = d_0, d_2, d_4, ...$ consists of bipolar pulses. This pulse stream is divided into an in-phase stream, $d_i(t)$ and a quadrature stream, $d_q(t)$ where:

$$d_i(t) = d_0, d_2, d_4, ... \quad \text{(even bits)}$$
$$d_q(t) = d_1, d_3, d_5, ... \quad \text{(odd bits)}$$

$d_i(t)$ and $d_q(t)$ each have half the bit rate of $d_k(t)$. A QPSK waveform $s(t)$ is constructed by amplitude modulating the in-phase and quadrature data streams onto the cosine and sine functions of a carrier wave according to the following formula [2]:

$$s(t) = \frac{1}{\sqrt{2}} d_i(t) \cos(2\pi f_c t + \frac{\pi}{4}) + \frac{1}{\sqrt{2}} d_q(t) \sin(2\pi f_c t + \frac{\pi}{4}) \quad (29)$$

The above formula can also be written as [2]:

$$s(t) = \cos[2\pi f_c t + \Theta(t)] \quad (30)$$

The pulse stream $d_i(t)$ amplitude modulates the cosine function with an amplitude of +1 or -1 which is the same as shifting the phase of the cosine wave by 0 or $\pi$. The result is a BPSK waveform. The pulse stream $d_q(t)$ also yields a BPSK wave after modulating the sine wave. The resulting waveform is orthogonal to the cosine function. The sum of these two functions outputs a QPSK waveform. The value of $\Theta(t)$ will correspond to one of four possible combinations of $d_i(t)$ and $d_q(t)$. These values are 0°, ±90° or 180°. In QPSK, the carrier phase can change only once every 2T. Thus the phase of the carrier during any $2T_b$ interval can be any one of the four phases corresponding to $\Theta(t)$. If in the next $2T_b$ interval, neither $d_i(t)$ or $d_q(t)$ changes sign, the carrier phase remains the same. If one of the pulse streams changes sign, a phase shift of ±90° results. Finally, if there is a sign change in both pulse streams, the carrier phase shifts 180°.
SIMULINK Implementation of a CDMA Transmitter

The model of the QPSK transmitter is shown in Figure 2.6

\[
\frac{1}{\sqrt{2}} \cos(\omega_s t + \pi / 4)
\]

\[
d_f(t) \times \sum \rightarrow s(t) = \cos(2\pi f_d t + \theta(t))
\]

\[
d_q(t)
\]

\[
\frac{1}{\sqrt{2}} \sin(\omega_s t + \pi / 4)
\]

Figure 2.6 QPSK Transmitter Model

2.8 Benefits of CDMA over TDMA and FDMA

Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA) are other types of multiple access methods. These techniques will be briefly discussed in sections 2.8.1 and 2.8.2. The advantages of CDMA over TDMA and FDMA will be discussed in section 2.8.3.

2.8.1 Frequency Division Multiple Access

In FDMA, specified subbands of frequency are allocated to users. The assignment of the user to the frequency band is long term or permanent. The communications resource can contain many spectrally separate signals. The first frequency band contains signals that operate between frequencies \( f_0 \) and \( f_1 \). The second band consists of frequencies between \( f_2 \) and \( f_3 \) and so on. There are buffer zones between frequency bands to reduce interference between adjacent
channels. These buffer zones are called ‘guard bands’. The major advantage of FDMA compared to TDMA is its simplicity. The FDMA channels do not require synchronisation or central timing. This is because each channel is almost independent of all other channels. A diagram illustrating FDMA is shown in Figure 2.7.

![Diagram of FDMA for three users](image)

**Figure 2.7 FDMA for three users**

2.8.2 Time Division Multiple Access

In TDMA, the communications resource is shared by assigning each user a specific time slot in the full bandwidth. The unused time regions between slots act as buffer zones to reduce interference. These zones are called guard times and they allow for some time uncertainty between signals in adjacent time slots. A diagram illustrating TDMA is shown in Figure 2.8.

![Diagram of TDMA](image)

**Figure 2.8 TDMA**
Figure 2.8 TDMA for one channel

For the comparison, Figure 2.9 illustrates CDMA. It can be seen that many users share the same channel bandwidth without the need for guard bands. Interference is avoided by using orthogonal PN sequences explained in section 1.5.

![Figure 2.9 CDMA illustrating three users transmitting over one channel](image)

2.8.3 Advantages of DS-CDMA

Some advantages of DS-CDMA are [9]:

1. **Specific selection capability.** A specific narrowband spectrum can be recovered from a spread spectrum with noise using CDMA. Other narrowband signals that have been spread could be part of the 'noise'. Thus, different signals can be recovered from the spread spectrum by despreading each signal with its own PN sequence. This is provided that the PN sequence is orthogonal.

2. **Allowance of multiple access using semi-orthogonal PN sequences.** Low channel interference is ensured even though more than one user is transmitting over the same spectrum. This is due to the low cross correlation property of these PN sequences.

3. **Signal hiding via low density power spectra.** The transmitted signal's energy spectrum is spread over a wide frequency. This allows the signal to be hidden by the channel noise which is at a higher power level. A 'secure' channel for transmission is thus achieved by preventing 'unauthorised' reading of the signal. The PN sequence also has the effect of 'scrambling'
the signal by adding an element of 'pseudorandomness' leading to further security.

4. **An equaliser is not needed.** In FDMA and TDMA, when the transmission rate is much higher than 10kbps, an equaliser is needed to reduce intersymbol interference (ISI). This ISI is caused by the spread in time delay. In CDMA, the receiver needs only a correlator to retrieve the desired signal. The correlator is usually much easier to implement than an equaliser.

5. **There is no guard time in CDMA.** TDMA requires guard time between time slots. Since the guard time is not used to transmit bits, this is a waste. These wasted bits could be used to improve the standard of performance of TDMA.

6. **CDMA is a natural waveform.** It is appropriate for microcell and in-building systems because it is susceptible to noise and interference.

7. **CDMA uses soft handoff.** There is no hard handoff from one frequency to another as the control of the signal is passed between cells. This is because every cell uses the same CDMA channel and the only difference is that PN sequences are assigned to the mobile terminals. Soft handoff will be explained in section 2.10.

8. **Synchronisation of the many communication sessions occurring at any given time on a LAN is not needed.** On a Local Area Network (LAN), each station is usually transmitting a segment of the time. The number of active stations at any given time is the measure of capacity for CDMA. It is not the total number of stations. Thus CDMA has an advantage because synchronisation is not necessary.

9. **Selective jamming or fading of the spread spectrum channel would only cause a small loss in the recovered signal’s spectral power.** This is because the signal is spread over a wide spectrum. If the power of the retrieved signal is above a certain threshold, no data is lost. Also, jamming effectiveness would be reduced because the jamming signal would have to be spread across this wide spectrum.

These advantages described above are either not available with FDMA and TDMA or are very hard to attain. For example, a narrowband communication
link that is able to tolerate multipath interference can be implemented by adding an adaptive equaliser in the receiver. However, the complexity of the receiver will also be increased and this may affect the ability to perform a smooth handover. No more than N users can simultaneously access a TDMA or FDMA system. If however, more than N users simultaneously access a CDMA system, the noise level and BER increase proportionally to the percent overload.

There are some disadvantages associated with CDMA. The two main limitations are "self jamming" and the near-far effect. Self jamming is caused by the spreading sequences not being orthogonal in an asynchronous CDMA network. This results in nonzero contributions to the user's test statistics when the signal is despread. In TDMA or FDMA, orthogonality can be secured for reasonable time or frequency guardbands. There are two main areas of concern for digital cellular radio. The first is multipath propagation. The received power falls off as the inverse of the distance between the transmitter and receiver raised to a power between two and four. Also the near-far problem is another cause for concern. This problem stems from the fact that in DS-CDMA, all the signals are transmitted on the same frequency band at the same time. This may result in the power of a nearby (unwanted) transmitted signal arriving at the listening receiver to overwhelm the signal from a distant (wanted) transmitter [19]. Hence power control techniques must be used to control the near-far effect. Power control and the near-far effect will be further discussed in section 2.9.

The final concern is the smooth handover from one cell to the next. This requires that the mobile acquires the new cell before it releases the old cell. Handoff is discussed in section 2.10.

2.9 Near-Far Problem

The near-far effect is sometimes called near-far interference and occurs when the receiver input includes one or more other CDMA signals that are stronger than the desired signal [20]. CDMA used to be rejected as unworkable in the mobile radio environment because of the 'near-far' problem. It was always
assumed that constant power was transmitted from all the stations. In a mobile radio environment, some users may be located near the base station while others are far away. These further users may experience propagation path losses in the range of many tens of dB.

2.9.1 Power Control

The near-far effect can be reduced by adapting the power of each transmitter to changes in the channel response or the interference environment. This adaptation is referred to as power control. The benefits of spreading are realised when the received powers from all users are approximately equal to each other rather than having constant power. Thus controlling the transmitter may result in equal received power.

2.9.2 Solution

Power control may be implemented by varying the transmitted power of the mobile units so that an adequate signal-to-interference ratio (SIR) is maintained at the receiver for each transmission [20]. Maximum capacity is achieved if the power control is adjusted so that the SIR is exactly what it needs to be for an acceptable error rate. The capacity and SNR have a reciprocal relationship [16].

2.10 Handoff

Handoff is the act of transferring support of a mobile from one base station to another. Using current technology, handoffs fail frequently, causing dropped calls. This results in poor service quality. Also, each handoff is followed and preceded by long periods of poor link quality which results in annoying noise and distortion. CDMA does not only reduce handoff failures, but also provides ‘soft handoff’. This maintains good voice quality at all times and the handoffs become undetectable even to skilled listeners [17]. Thus CDMA handoff differs from normal standards in many aspects [17]:
SIMULINK Implementation of a CDMA Transmitter

- It is 'soft' which means that the handoff does not interrupt communication.
- The handoff is not abrupt but is rather a prolonged call state during which there is communication via two or more base stations. The link performance during handoff is improved by the multi-way communication diversity. This diversity gain also partly compensates for the large path loss at the cell boundary.
- The signal measurement that triggers the handoff is performed by the mobile stations, not the base stations.

There is no handoff boundary in CDMA, but a handoff region instead. The handoff can be completed either by the mobile moving completely into the new cell or by the mobile going back to the original serving cell. A call is never in jeopardy due to link failure in both cases.

2.11 Multiuser Detection

As has been described, multiple access allows multiple users to share moderate capacity resources such as bandwidth and time. In a conventional DS-CDMA system, each user is treated separately as a signal. The other users are considered as interference or noise. The interference suppression capability is measured by the processing gain. This suppression capability has limitations and as the number of interfering users increases, the BER also increases. Also, even if the number of users is still not large, some users may be received at such high levels that a user transmitting at a lower power may be drowned out. This is the near-far effect and this has been described in section 2.9. The recent interest in DS-CDMA has been due to the fact that tight power control has been implemented successfully.

In a CDMA system, all users interfere with each other. Potential capacity increases can theoretically be achieved if the negative effect that each user has on others can be eradicated. This is multiuser detection in which all users are considered as signals for each other. So instead of users interfering with each other, they are being used for their shared advantage by joint detection.
Optimal multiuser detection has high complexity so sub-optimum detectors are considered.

In a cellular system, many mobiles communicate with one Base Station (BS). The BS has to detect all the signals while each mobile is only concerned with its own signal. The BS has knowledge of the PN sequences of all its mobiles. Thus multiuser detection is directed mainly at the BS or in the reverse link (mobile to BS) [15]. One of the issues of mobile systems pertinent for multiuser detection is multipath and this will be discussed in section 2.12.

### 2.12 Multipath Channels

A multipath channel has multiple propagation paths. That is, there is more than one path from the transmitter to the receiver. Multipath is caused in free space propagation by reflections from objects in the surroundings. It may also be caused by atmospheric refraction or by multiple reflection layers in the ionosphere for some carrier frequencies. This might produce fluctuations in the received signal level. Multipath is normal in telephone circuits and other two-way communication systems. In the telephone circuit, echoes are caused by unintentional coupling between the receiver and the transmitter. The different paths may be made up of many distinct paths. Each path has a different time delay and attenuation. On the other hand, the different paths might consist of non-discrete paths. The multipath wave is delayed by a certain time \( \tau \) compared to the wave on the direct path. In the DS SS system, if we assume that the receiver is synchronised to the RF phase of the direct path or the time delay, the received signal can be expressed as [2]:

\[
    r(t) = Ab(t)g(t)\cos \omega_d t + \alpha Ab(t-\tau)g(t-\tau)\cos (\omega_d t + \theta) + n(t),
\]

(31)

where \( b(t) \) is the data signal, \( g(t) \) is the code signal, \( n(t) \) is a zero-mean Gaussian noise process and \( \tau \) is the differential time delay between the two paths in the interval \( 0 < \tau < T \). The attenuation of the multipath signal relative to the direct path signal is \( \alpha \) and \( \theta \) is a random phase in the range of \( (0, 2\pi) \). For the receiver that is synchronised with the direct path signal, the correlator output \( z(t) \) at time \( t = T \) is [2]:
where $g^2(t) = 1$. For codes with long periods where $\tau > T_e$, $g(t)g(t-\tau) \approx 0$. Hence, if the chip duration $T_c$ is less than the differential time delay between the multipath and direct path signals, the output of the correlator becomes:

$$z(t) = \int_0^T [2Ab(t)\cos\omega_d t + 2n(t)g(t)\cos\omega_d t] dt$$

$$= Ab(T) + n_o(t),$$

where $n_o(T)$ is a zero-mean Gaussian random variable. With the code-correlation receiver, the spread spectrum CDMA system can eradicate the interference caused by multipath. However, with shorter PN sequences, problems may still arise due to multipath. This will be discussed in section 2.13.

### 2.12.1 Multipath Fading

Multipath fading is due to the superposition of the different multipath signals. It can result in attenuation of the signal that is frequency dependent. This phenomenon is known as frequency selective fading [18]. Refer to Figure 2.10 for the illustration of how reflections of a signal can lead to multipath fading [18].

![Figure 2.10 Illustration of How a Reflection of a Signal Results in Different Propagation Delays [18.](image-url)]
Since the receiver antenna intercepts the superimposed signals, the worst case scenario is when path A entirely cancels out path B (the original signal). In a multipath channel, the received signal strength varies considerably with time because of the changing relationship between multiple propagation paths. Thus for a slowly fading channel, the signal strength change is slow in relation to the symbol rate. The received signal can be represented as [4]:

\[ v(t) = a(t)e^{j\theta(t)}c(t) + n(t), \]  

where \( a(t) \) is a random variable with mean equal to 1, \( n(t) \) is white noise and \( \theta(t) \) is a random variable denoting the phase error. The variables \( \theta(t) \) and \( a(t) \) vary slowly in comparison to \( c(t) \).

Fading can also be characterised as being Rayleigh or Rician. Rayleigh fading is the result of a vector sum of multiple signal components. Each of these signal components has a random amplitude. It can also be viewed as a signal whose in-phase and quadrature components are Gaussian random variables. Rayleigh fading causes deep signal dropouts.

The probability density function (pdf) of Rayleigh fading is given by [21]:

\[ Pr(r) = \frac{r}{\sigma^2}e^{-\frac{r^2}{2\sigma^2}}, \quad r \geq 0, \]  

where \( \sigma \) is the Rayleigh parameter (the most probable value). The mean and the variance of this distribution is \( \sigma \sqrt{\frac{\pi}{2}} \) and \( (2 - \pi/2)\sigma^2 \), respectively.

The fading is said to be Rician if there is a strong constant signal component in addition to the multiple random components of Rayleigh fading. The strong component may be a line of sight path or a path that goes through much less attenuation compared to other arriving components [21]. When such a strong path exists, the received signal can be considered to be the sum of two vectors: a scattered Rayleigh vector with random amplitude and phase and a vector which is deterministic in amplitude and phase, representing the strong component. Rician fading is typical of situations where there is a direct,
unobstructed path between stations, as well as reflecting or scattering surfaces [16].

The Ricean fading has a distribution given by the pdf [21]:

\[ Pr(r) = \frac{r}{\sigma^2} \exp \left( -\frac{r^2 + \nu^2}{2\sigma^2} \right) I_0 \left( \frac{\nu r}{\sigma^2} \right), \quad r \geq 0, \]  

(36)

where \( I_0 \) is the 0-th order Bessel function of the first kind, \( \nu \) is the magnitude (envelope) of the strong component and \( \sigma^2 \) is proportional to the power of the "scatter" Rayleigh component [21].

2.13 Intersymbol Interference and Multiple Access Interference

2.13.1 Intersymbol Interference

In a typical digital baseband system, there are filtering aspects with circuit reactances in the transmitter, receiver and channel. The input pulses may be flat top or impulse-like. In both cases, the channel reactances can distort the amplitude and phase of the pulses. At the transmitter, the pulses are low-passed filtered to restrict them to certain bandwidth. The receiving filter is called an equalizing filter and should be adjusted to make up for the distortion caused by the transmitter and channel. A transfer function that combines the effects of all this filtering is as follows [2]:

\[ H(f) = H_t(f) H_c(f) H_r(f), \]  

(37)

where \( H_t(f) \) represents the filtering at the transmitter, \( H_c(f) \) is the filtering in the channel and \( H_r(f) \) is the filtering done at the receiver.

If these pulses are improperly filtered as they pass through the communication system, they will be overlapped when received. The pulse of one symbol "smears" into neighbouring time slots and this interferes with the detection process. This interference is called Intersymbol Interference (ISI). In radio communications, ISI is mainly caused by multipath propagation leading to the delayed version of the signal extending into the next sampling interval.
2.13.2 Multiple Access Interference

It has been shown that interference between multiple users can be avoided if the codes used to transmit the data are orthogonal. However, in reference to terminal to base station (BS) transmission, if the delays between the transmitter and receiver are different, the signals are not considered orthogonal. In a 50m coverage area, depending on the data rate, these delay differences may be around a few chips. This effect is referred to as Multiple Access Interference (MAI) [3]. MAI is quite serious for very short spreading sequences such as 7-bits Gold codes and 16-bits Walsh-Rademacher functions. These short PN sequences also cause problems with multipath propagation.

2.14 Reduction Of ISI and MAI

In Asynchronous Transfer Mode Wireless Local Area Networks (ATM WLANs), a data rate of several Mbps is required. For ATM WLANs, there is a bit error rate (BER) degradation that is caused by the dispersion of the channel on account of multipath propagation. In DS CDMA, the signal intercepted at the receiver without taking different path losses into consideration is [3]:

\[ R(t) = g_1(t - \tau_1)s_1(t - \tau_1) + g_2(t - \tau_2)s_2(t - \tau_2) + \ldots + g_N(t - \tau_N)s_N(t - \tau_N), \]

where \( \tau_i \); \( i = 1, 2, \ldots, N \) are delays corresponding to different transmission paths associated with the \( i \)-th user.

If the receiver is only to accept messages from transmitter (user) one, the receiver has been perfectly synchronised with that user. Hence the signal \( r(t) \) obtained is [3]:

\[ r(t) = g_1^2(t - \tau_1)s_1(t - \tau_1) + \ldots + g_1(t - \tau_1)g_N(t - \tau_N)s_N(t - \tau_N), \]

where \( g_1^2(t - \tau_1)s_1(t - \tau_1) \) is the signal that is needed and the other terms \( g_1(t - \tau_1)g_i(t - \tau_N)s_i(t - \tau_N); \ (i \neq 1) \) are the interfering signals that cause MAI. The MAI becomes more severe as the correlation between the code \( g_1(t - \tau_1) \) and the other codes \( g_i(t - \tau_i) \) gets stronger. This is because the signal is finally demodulated using cross correlators or a matched filter.
### 2.14.1 Multipath in DS-CDMA

The received signal where there are different signal delays due to multipath can be described as [3]:

\[ R(t) = A_1 g_1(t - \tau_1) s_1(t - \tau_1) + A_2 g_2(t - \tau_2) s_2(t - \tau_2) + \ldots + A_M g_M(t - \tau_M) s_M(t - \tau_M), \]  

(40)

where the coefficients \( A_1 > A_2 > \ldots > A_M \) are the amplitudes of the signals with different propagation paths. The strongest signal component \( A_1 g_1(t - \tau_1) s_1(t - \tau_1) \) is synchronised with the receiver. The other terms have delays that are not equal to \( \tau_i \) (\( i \neq 1 \)). Hence these terms cause ISI.

To reduce this ISI, the codes \( g_1(t)\ldots g_M(t) \) are improved so that for delays larger than a single chip, they have a low auto-correlation. Gold codes are a set of these improved codes. However, for shorter Gold sequences, their auto-correlation still possesses significant magnitude of the order of 0.71 [3]. In order to decrease both the cross- and auto-correlation functions, the carrier waveform must be modified to produce optimum values [3].

### 2.14.2 Carrier Waveform Modification

According to the proposed method in [3], the autocorrelation and cross correlation functions can be decreased by modifying the carrier waveform so that its period \( T_M > 25T_c \). Please note that this value is only an example. The magnitude of the autocorrelation function must be much lower than one for all values of the delays \( T_c < \tau \leq 25T_c \). By the inclusion of an additional frequency modulation (FM) with the existing modulating function, this can be achieved. The modulating function has a period of \( T_M \). To simplify the detector, \( T_M \) is chosen to be an integer multiple of the PN code length. Where [3]:

\[ T_M = k \times 7T_c; \quad k = 1, 2, 3, \ldots \]

The function used in this project that fulfils these conditions is:
SIMULINK Implementation of a CDMA Transmitter

\[ W(t) = \frac{2}{9} \alpha \left[ Tr\left( \frac{t}{7T_c} \right) + \beta Tr\left( \frac{t}{14T_c} \right) \right] \quad (41) \]

where

\[ Tr(t) = \begin{cases} 
-2(t-0.5), & 0 \leq t < 1 \\
2(t-1.5), & 1 \leq t < 2 
\end{cases} \quad (42) \]

is of period 2 and is a periodic function. The values of \( \alpha \) and \( \beta \) used in this project are 3.72 and 0.2 respectively.
3 SIMULINK Implementation of a CDMA Transmitter

3.1 Proposed Transmitter Model

The proposed transmitter model for this project is shown in Figure 3.1.

Using MATLAB/SIMULINK, a CDMA transmitter incorporating BPSK modulation was designed. This design was also modified to accommodate QPSK modulation. A data sequence comprising 16 bits was used. This number can later be modified to accommodate sequences of any number of bits. The set of PN sequences was comprised of 7-bit Gold codes. This set of PN sequences allows a maximum of 9 users. The PN sequence length can also be changed if needed later on. The SIMULINK model of the transmitter will be discussed in more detail in sections 3.3 and 3.4.

3.2 SIMULINK Overview

SIMULINK is a program for simulating dynamic systems. It is a visual extension of MATLAB with many additional features specific to dynamic systems while retaining all of MATLAB’s general purpose functionality [11].

A visual interface is supplied by SIMULINK where a system can be designed from either in-built or user-defined blocks. The blocks are connected by
drawing signal lines or by using special interconnecting blocks. This creates a block diagram representing the system. The simulation can be performed and monitored. The monitoring can be done via scopes that are connected to the relevant points on the signal line. On the other hand, the simulation results can be viewed in MATLAB by using the workspace variables to store the results. By typing in the workspace variable name in the MATLAB environment, the stored contents can be viewed. Alternatively, the results can be plotted to a graph. The simulation can also be paused, so that the performance of the system can be evaluated, and then restarted from the paused state.

The in-built blocks on SIMULINK are available from a template library. The library contains a vast selection of blocks from different categories. These blocks can represent analogue circuits, digital circuits, filters, scopes, memory, logic functions, hardware connections and many more. The desired blocks can be selected and then connected to make the system model. Each block has specific parameters that can be set by the designer.

There are many advantages incurred by using SIMULINK. An outline of these is given below [12].

3.2.1 Perceptiveness

SIMULINK allows the user to model less complex systems without programming. By the use of a visual design, the user can use intuition to build the system using blocks. The block diagram is a realistic and familiar form to engineers as a way to represent a system. Thus, better understanding of the system behaviour arises as a result.

3.2.2 Convenience

Pre-built blocks and MATLAB function libraries are conveniently offered by SIMULINK. Designing the system is of paramount importance to engineers rather than constructing analysis tools and modelling components. Most of the basic building blocks and analysis tools needed for electronic systems are provided by the SIMULINK and MATLAB toolbox libraries. Since the designer does not have to write any code to perform the simulation, it can be said that
SIMULINK provides the platform for which systems can be simulated. Thus the user is only involved with the modelling of the system.

3.2.3 Flexibility
Monitoring the progress of the simulation is very easy in SIMULINK. Observation of the signal flow through the lines connecting the blocks can be done via different kinds of probes. Also, the system variables can be analysed by pausing the simulation and then restarting from where it left off.

3.2.4 Modularity
The system design approach in SIMULINK can be described as object-oriented. The blocks (objects) communicate by sending signals to one another and hence are separate entities. Procedural language, on the other hand has procedures which communicate via shared data structures. Thus it is easy to modify the system, if the need be, using SIMULINK due to the high coherence of the blocks and low affiliation between them.

3.2.5 Power of procedural language
Sometimes it is best to combine MATLAB function blocks together with SIMULINK blocks. These circumstances may arise when the system to be modelled is large. This is because non-linear blocks that are very big and complex may have to be used. By combining these two types of blocks, the power of both MATLAB and SIMULINK's procedural language can be exploited. Vector processing can be implemented very conveniently using MATLAB's programming language. It is also very convenient because variables and data structures do not have to be initialised.

3.2.6 S-functions
When a SIMULINK model is created, a new function called an S-function (System-Function) becomes available in MATLAB [11]. The dynamics of the model are defined by this function. The S-function has the calling syntax: 
\[ sys = \text{model}(t, x, u, \text{flag}) \]
where *model* is the model name, and *flag* controls the information returned in *sys*. For example, a *flag* set to 1 gives the state derivatives in the variable *sys* at the operating point defined by the time *t*, input vector *u* and state vector *x*. The S-function is used by the linearization, trim and integration routines to determine the dynamics of the system. It behaves like any other MATLAB function and has the following benefits [11]:

- You can create the linear or non-linear model in many different languages such as block diagrams or M-files.
- You can create new types of blocks that can be used in any block diagram.
- You can write your own analysis and simulation routines.

Thus S-functions are simply MATLAB functions with a special calling syntax which allows you to access the dynamic equations of a model.

### 3.2.7 Combining MATLAB Functions With SIMULINK

SIMULINK can model analogue and digital, linear and non-linear systems. Usually, differential or difference equations are used to model systems. State space equations can be implemented by utilising user defined blocks. The existing state space function block in SIMULINK can be used if the block to be defined is linear. If the block is non-linear, it may be defined using a text editor since S-functions are stored away. State space function blocks can be used directly whereas S-functions must utilise S-function blocks.

You may also define non-linear blocks using the pre-built MATLAB function block. The MATLAB function can then be implemented in SIMULINK. This allows very large and complex blocks to be created. However, the MATLAB function block is limited to MATLAB functions only. The user defined blocks in MATLAB are created using S-functions which apply the technique of state space equations.
3.3 SIMULINK Implementation of a CDMA Transmitter Using BPSK

For the data sequence, a ‘Clock’ block multiplied by a ‘Constant’ block with the value of 1 is used to generate a signal. This signal is then put through a ‘Fcn’ block with the MATLAB function ‘rem’. This finds the remainder of the signal after it is divided by 16. This remainder is put through a MATLAB ‘Fcn’ block using the function ‘fix’ to generate 16 discrete levels. These 16 discrete levels ranged from 0 to 15. The 16 levels were then mapped onto the 16 data bits using a lookup table. For simulation purpose, a ‘2-d lookup table’ was used to because there were nine sets of data sequences corresponding to the nine users. Hence it made things much easier because to choose a data sequence, it was simply a matter of entering the user number into the ‘Constant’ block connected to the X Index of the ‘2-d lookup table’. The resulting waveform was multiplied by \( \cos(W(t) + 2\pi f_W(t) \, dt) \) where:

\[
W(t) = \frac{2}{\alpha} \left[ T_r \left( \frac{t}{7T_c} \right) + \beta T_r \left( \frac{t}{14T_c} \right) \right]
\]

to complete the BPSK modulation. The PN sequence was generated in the same way as the data sequence except that the ‘Constant’ block had the value of 7 instead of 1. This was to ensure that one bit of data was multiplied by the whole 7-bit PN sequence. Also, the signal was divided by 7 and the remainder found and fixed to generate 7 discrete levels from 0 to 6. The 7 levels were mapped onto the 7-bit Gold sequence using a ‘2-d lookup table’. This was so that a particular PN sequence corresponding to one of the nine users could be chosen. The resulting PN sequence was then multiplied by the BPSK modulated waveform to complete the transmitter. The transmitter ensures that each bit of the data sequence is multiplied by the whole PN sequence.
The $W(t)$ function was implemented as Figure 3.2:

Figure 3.2 SIMULINK Implementation of the Function $W(t)$

Firstly the $T_r\left(\frac{t}{7T_c}\right)$ function was constructed.

The model is shown in Figure 3.3:

Figure 3.3 SIMULINK Implementation of the Function $\pi\left(\frac{t}{T_c}\right)$

It consists of a 'Clock' block multiplied by a 'Gain' block that contains the value $\pi$. This is because it is now multiplied by a MATLAB 'Fcn' block containing a 'sawtooth' waveform that has a period of $2\pi$. Hence the resulting waveform has a period of 2 since $T_c = 1/7$ and $7*T_c = 1$. The absolute value is taken using the 'Abs' block and the resulting function is multiplied by 2 and has 1 subtracted from it.
Next the \( Tr\left( \frac{t}{14T_c} \right) \) function is constructed in the same way except that the clock is multiplied by \( \pi/2 \) instead of \( \pi \). This is because \( 14 \cdot T_c = 2 \). Hence the resulting waveform has a period of 4 since the sawtooth waveform has a period of \( 2\pi \).

The \( Tr\left( \frac{t}{7T_c} \right) \) function is then multiplied by a 'Gain' block with the value \( 2/9\alpha \) where \( \alpha \) is 3.72. The \( Tr\left( \frac{t}{14T_c} \right) \) function is multiplied by a 'Gain' block with the value \( 2/9\alpha \beta \) where \( \beta \) has the value 0.2. These two values are then summed and the integral is taken using the 'Integrator' block. The resulting value is then multiplied by \( 2\pi \), added to \( W_c \) and the cosine of this total value is taken using a MATLAB 'Fcn' block with the 'cos' function.

A diagram of the SIMULINK implementation of a CDMA transmitter using BPSK is shown in Figure 3.4.
Figure 3.4 SIMULINK Implementation of a CDMA Transmitter Using BPSK
3.4 SIMULINK Implementation of a CDMA Transmitter Using QPSK

The QPSK modulated transmitter was implemented in much the same way as the BPSK version except for a few minor changes. The 'Clock' block multiplied by the 'Constant' block with the value of 1 was still used to generate the signal. The 16 discrete levels were simultaneously mapped onto 16 even and odd numbered levels using one 'look up table' for the even levels and another 'lookup table' for the odd levels. These even and odd levels were then mapped onto the even and odd data bits of a 16 bit data sequence also using a lookup table. Again for fast simulation purposes, a '2-d lookup table' was used to choose the appropriate data sequence corresponding to a particular user. This design can also be modified to accommodate 9 to 16 bit sequences.

The even data bits were multiplied by the function $\cos(W_c t + \frac{2\pi}{T} \int W(t) \, dt)$. The odd data bits were multiplied by the function $\sin(W_c t + \frac{2\pi}{T} \int W(t) \, dt)$ which was constructed in exactly the same way as the function $\cos(W_c t + \frac{2\pi}{T} \int W(t) \, dt)$ except that the 'sin' function was used in the 'Fcn' block instead of the 'cos' function. The resulting waveforms were then summed to complete the QPSK modulation. The QPSK waveform was then multiplied by the PN sequence to complete the transmitter.

The PN sequence was generated in the same way as for BPSK except that the 'Clock' block was multiplied by a 'Constant' block with the value of 7/2. This was to ensure that one period of the QPSK waveform is multiplied by the whole PN sequence. Refer to Figure 3.5 for a diagram of the SIMULINK implementation of a CDMA transmitter using QPSK.
Figure 3.5 SIMULINK Implementation of a CDMA Transmitter Using QPSK.
SIMULINK Implementation of a CDMA Transmitter

4 Simulation Trials

4.1 Results Using BPSK Modulation

The following tables outline the values that were used in each simulation. The PN Sequences used are an optimum set of Gold code sequences. These Gold code sequences are 7-bits in length and can accommodate up to 9 users. The users and their specified 7-bit Gold code sequences are shown in the following Table 4.1-1 [18].

The data sequences used were 16 bits long and were randomly chosen. These data sequences are given in Table 4.1-2.

<table>
<thead>
<tr>
<th>User, k</th>
<th>7-bit Gold Code Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1-1-1-1-1-1-1</td>
</tr>
<tr>
<td>1</td>
<td>11-1-1-1-1</td>
</tr>
<tr>
<td>2</td>
<td>-1 1 1 -1 1</td>
</tr>
<tr>
<td>3</td>
<td>-1 1 1 1 -1 -1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 -1 1 1 1</td>
</tr>
<tr>
<td>5</td>
<td>1 -1 -1 -1 -1 1</td>
</tr>
<tr>
<td>6</td>
<td>-1 1 1 -1 1 1 1</td>
</tr>
<tr>
<td>7</td>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>8</td>
<td>-1 -1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

Table 4.1-1 CDMA 7-bits Gold Code Set

For each simulation, a matrix of 100 timesteps was used to capture:

1. The data after BPSK modulation.
2. The PN sequence.
3. The transmitter output signal.

The MATLAB 'plot' command was used to graph the data.

A total of 9 simulations were conducted corresponding to 9 users.

Notice that in each simulation, it can be seen that one QPSK symbol is multiplied by one 7-bit PN sequence. The resulting waveform is shown in the
transmitter output. The plots of the data, PN sequence and transmitter output signal for simulations 1 to 9 can be seen in Figures 4.1-2 - 4.1-10

<table>
<thead>
<tr>
<th>Simulation number</th>
<th>Data Sequence</th>
<th>PN Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>1-1-1-1-1-1</td>
</tr>
<tr>
<td>2</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>1-1-1-1-1-1</td>
</tr>
<tr>
<td>3</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>-1-1-1-1-1-1</td>
</tr>
<tr>
<td>4</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>-1-1-1-1-1-1</td>
</tr>
<tr>
<td>5</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>1-1-1-1-1-1</td>
</tr>
<tr>
<td>6</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>1-1-1-1-1-1</td>
</tr>
<tr>
<td>7</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>-1-1-1-1-1-1</td>
</tr>
<tr>
<td>8</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>1-1-1-1-1-1</td>
</tr>
<tr>
<td>9</td>
<td>1-1-1-1-1-1-1-1-1-1-1-1-1-1-1</td>
<td>-1-1-1-1-1-1</td>
</tr>
</tbody>
</table>

Table 4.1-2 Data Sequences Used in Each Simulation.

Figure 4.1-1 is a graph of one period of $W(t)$ from the simulation.

![A Period of the Function $W(t)$](image)

Figure 4.1-1 A Period of the Function $W(t)$
Figure 4.1-2 Simulation 1
Figure 4.1-3 Simulation 2
SIMULINK Implementation of a CDMA Transmitter

Data After BPSK Modulation

PN Sequence [-1 1 1 -1 1 -1 1 1]

Transmitter Output

Figure 4.1-4 Simulation 3
SIMULINK Implementation of a CDMA Transmitter

Figure 4.1-5 Simulation 4
Figure 4.1-6 Simulation 5
SIMULINK Implementation of a CDMA Transmitter

Data After BPSK Modulation

PN Sequence [1 -1 -1 -1 -1 -1]

Transmitter Output

Figure 4.1-7 Simulation 6
**Figure 4.1-8 Simulation 7**
SIMULINK Implementation of a CDMA Transmitter

Figure 4.1-9 Simulation 8
Figure 4.1-10 Simulation 9
4.2 Results Using QPSK Modulation

For each simulation, a matrix of 100 timesteps was used to capture:
1. The data after QPSK modulation.
2. The PN sequence.
3. The transmitter output signal.

The MATLAB 'plot' command was used to graph the above data.
A total of 9 simulations were conducted corresponding to 9 users.

Table 4.2-1 outlines which values were used in each simulation.

<table>
<thead>
<tr>
<th>Simulation number</th>
<th>Data Sequence</th>
<th>PN Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-1 1-1 1-1-1 1-1-1 1-1-1-1 1-1-1-1-1</td>
<td>1-1 1-1-1-1</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1 1-1-1 1-1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>1 1 1-1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>1-1 1 1-1-1-1 1 1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>-1 1 1-1 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 1-1-1-1-1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>-1 1 1-1 1 1</td>
</tr>
<tr>
<td>5</td>
<td>1-1 1 1-1 1 1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>1 1-1 1 1 1</td>
</tr>
<tr>
<td>6</td>
<td>-1-1 1-1-1-1 1 1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>1-1 1-1 1 1</td>
</tr>
<tr>
<td>7</td>
<td>-1 1 1-1-1-1 1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>-1 1 1-1 1 1</td>
</tr>
<tr>
<td>8</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>9</td>
<td>-1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1-1</td>
<td>-1 1-1 1 1 1</td>
</tr>
</tbody>
</table>

Table 4.2-1 Data Sequences Used in Each Simulation.

Notice that in each simulation, it can be seen that one QPSK symbol is multiplied by one 7-bit PN sequence. The resulting waveform is shown in the transmitter output. The plots of the data, PN sequence and transmitter output signal for simulations 1 to 9 can be seen in Figures 4.2-1 to 4.2-9.
SIMULINK Implementation of a CDMA Transmitter

Figure 4.2-1 Simulation 1
Figure 4.2-2 Simulation 2
SIMULINK Implementation of a CDMA Transmitter

Data After QPSK Modulation

PN Sequence [-1, 1, -1, 1, -1, 1]

Transmitter Output

Figure 4.2-3 Simulation 3
SIMULINK Implementation of a CDMA Transmitter

Figure 4.2-4 Simulation 4
SIMULINK Implementation of a CDMA Transmitter

Figure 4.2-5 Simulation 5
SIMULINK Implementation of a CDMA Transmitter

Data After QPSK Modulation

PN Sequence [1 -1 -1 -1 -1 -1 1 1]

Transmitter Output

Figure 4.2-6 Simulation 6
Figure 4.2-7 Simulation 7
Figure 4.2-8 Simulation 8
SIMULINK Implementation of a CDMA Transmitter

Data After QPSK Modulation

PN Sequence [-1 -1 -1 1 1 1]

Transmitter Output

Figure 4.2-9 Simulation 9
4.3 Cross Correlation Values

The cross correlation coefficients of the set of PN sequences are given in Table 4.3-1.

| 1.0000 | -0.1667 | -0.1667 | -0.0913 | -0.1667 | -0.0913 | -0.3536 | -0.1667 |
| -0.1667 | 1.0000 | -0.1667 | 0.5477 | -0.1667 | 0.5477 | -0.7303 | 0.4714 | -0.1667 |
| -0.1667 | -0.1667 | 1.0000 | -0.0913 | -0.1667 | -0.0913 | -0.0913 | -0.3536 | -0.1667 |
| -0.0913 | 0.5477 | -0.0913 | 1.0000 | -0.0913 | -0.4000 | -0.4000 | 0.2582 | -0.0913 |
| -0.1667 | -0.1667 | -0.1667 | -0.0913 | 1.0000 | -0.0913 | -0.0913 | -0.3536 | -0.1667 |
| -0.0913 | 0.5477 | -0.0913 | -0.4000 | -0.0913 | 1.0000 | -0.4000 | 0.2582 | -0.0913 |
| -0.0913 | -0.7303 | -0.0913 | -0.4000 | -0.0913 | -0.4000 | 1.0000 | 0.2582 | -0.0913 |
| -0.3536 | 0.4714 | -0.3536 | 0.2582 | -0.3536 | 0.2582 | 0.2582 | 1.0000 | -0.3536 |
| -0.1667 | -0.1667 | -0.1667 | -0.0913 | -0.1667 | -0.0913 | -0.0913 | -0.3536 | 1.0000 |

Table 4.3-1: Cross Correlation Coefficients of the Set of PN Sequences (7-bit Gold Codes)

The cross correlation coefficients of the 9 PN sequences used in the BPSK and QPSK simulations are shown in Table 4.3-2 and Table 4.3-5 respectively. The same data sequence [1 1 1 1 1 1 1 1 1 1 1] was used in all nine simulations.

The cross correlation coefficients of the 9 output signals using BPSK modulation and the \( W(t) \) function are shown in Table 4.3-3. The values of the cross correlation coefficients of the same simulation but without the \( W(t) \) function are shown in Table 4.3-4.

For QPSK modulation, the cross correlation coefficients of the 9 output signals using the \( W(t) \) function are shown in Table 4.3-6. The cross correlation coefficient values of the same simulation but without the \( W(t) \) function are shown in Table 4.3-7.
| 1.0000 | -0.1658 | -0.1658 | -0.0921 | -0.1682 | -0.0896 | -0.0921 | -0.3533 | -0.1658 |
| -0.1658 | 1.0000 | -0.1675 | 0.5486 | -0.1658 | 0.5472 | -0.7302 | 0.4713 | -0.1675 |
| -0.1658 | -0.1675 | 1.0000 | -0.0908 | -0.1658 | -0.0928 | -0.0908 | -0.3541 | -0.1675 |
| -0.0921 | 0.5486 | -0.0908 | 1.0000 | -0.0921 | -0.3996 | -0.4006 | 0.2585 | -0.0908 |
| -0.1682 | -0.1658 | -0.1658 | -0.0921 | 1.0000 | -0.0896 | -0.0921 | -0.3533 | -0.1658 |
| -0.0896 | 0.5472 | -0.0928 | -0.3996 | -0.0896 | 1.0000 | -0.3996 | 0.2579 | -0.0928 |
| -0.0921 | -0.7302 | -0.0908 | -0.4006 | -0.0921 | -0.3996 | 1.0000 | 0.2585 | -0.0908 |
| -0.3533 | 0.4713 | -0.3541 | 0.2585 | -0.3533 | 0.2579 | 0.2585 | 1.0000 | -0.3541 |
| -0.1658 | -0.1675 | -0.1675 | -0.0908 | -0.1658 | -0.0928 | -0.0908 | -0.3541 | 1.0000 |

Table 4.3-2: Cross Correlation Coefficients of the 9 PN Sequences for the BPSK Simulation

| 1.0000 | -0.1421 | -0.1464 | -0.1432 | -0.1395 | -0.1419 | -0.1424 | -0.1416 | -0.1451 |
| -0.1421 | 1.0000 | -0.1429 | 0.4241 | -0.1414 | 0.4296 | -0.7155 | 0.4309 | -0.1426 |
| -0.1464 | -0.1429 | 1.0000 | -0.1419 | -0.1451 | -0.1432 | -0.1416 | -0.1424 | -0.1395 |
| -0.1432 | 0.4241 | -0.1419 | 1.0000 | -0.1424 | -0.1464 | -0.1395 | -0.1451 | -0.1416 |
| -0.1395 | -0.1414 | -0.1451 | -0.1424 | 1.0000 | -0.1416 | -0.1432 | -0.1419 | -0.1464 |
| -0.1419 | 0.4296 | -0.1432 | -0.1464 | -0.1416 | 1.0000 | -0.1451 | -0.1395 | -0.1424 |
| -0.1424 | -0.7155 | -0.1416 | -0.1395 | -0.1432 | -0.1451 | 1.0000 | -0.1464 | -0.1419 |
| -0.1416 | 0.4309 | -0.1424 | -0.1451 | -0.1419 | -0.1395 | -0.1464 | 1.0000 | -0.1432 |
| -0.1451 | -0.1426 | -0.1395 | -0.1416 | -0.1464 | -0.1424 | -0.1419 | -0.1432 | 1.0000 |

Table 4.3-3: Cross Correlation Coefficients of the 9 Output Signals Using BPSK With the W(t) Function

| 1.0000 | -0.1451 | -0.1411 | -0.1411 | -0.1451 | -0.1451 | -0.1411 | -0.1451 | -0.1411 |
| -0.1451 | 1.0000 | -0.1412 | 0.4315 | -0.1451 | 0.4274 | -0.7137 | 0.4274 | -0.1411 |
| -0.1411 | -0.1412 | 1.0000 | -0.1451 | -0.1411 | -0.1411 | -0.1451 | -0.1411 | -0.1451 |
| -0.1411 | 0.4315 | -0.1451 | 1.0000 | -0.1411 | -0.1411 | -0.1451 | -0.1411 | -0.1451 |
| -0.1451 | -0.1451 | -0.1411 | -0.1411 | 1.0000 | -0.1411 | -0.1451 | -0.1411 | -0.1451 |
| -0.1451 | 0.4274 | -0.1411 | -0.1411 | -0.1451 | 1.0000 | -0.1411 | -0.1451 | -0.1411 |
| -0.1411 | -0.7137 | -0.1451 | -0.1451 | -0.1411 | -0.1411 | 1.0000 | -0.1411 | -0.1451 |
| -0.1451 | 0.4274 | -0.1411 | -0.1411 | -0.1451 | -0.1451 | 1.0000 | -0.1411 | -0.1451 |
| -0.1411 | -0.1411 | -0.1451 | -0.1451 | -0.1411 | -0.1451 | -0.1451 | 1.0000 | -0.1411 |

Table 4.3-4: Cross Correlation Coefficients of the 9 Output Signals Using BPSK Without the W(t) Function
### Table 4.3-5: Cross Correlation of the 9 PN Sequences for the QPSK Simulation

<table>
<thead>
<tr>
<th></th>
<th>1.0000</th>
<th>-0.1469</th>
<th>-0.1431</th>
<th>-0.1395</th>
<th>-0.1428</th>
<th>-0.1467</th>
<th>-0.1380</th>
<th>-0.1456</th>
<th>-0.1437</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>-0.1469</td>
<td>1.0000</td>
<td>-0.1393</td>
<td>0.4277</td>
<td>-0.1455</td>
<td>0.4286</td>
<td>-0.7151</td>
<td>0.4286</td>
<td>-0.1381</td>
</tr>
<tr>
<td>-0.1469</td>
<td>1.0000</td>
<td>-0.1393</td>
<td>1.0000</td>
<td>-0.1468</td>
<td>-0.1437</td>
<td>-0.1395</td>
<td>-0.1456</td>
<td>-0.1380</td>
<td>-0.1428</td>
</tr>
<tr>
<td>-0.1437</td>
<td>-0.1393</td>
<td>1.0000</td>
<td>-0.1468</td>
<td>1.0000</td>
<td>-0.1381</td>
<td>-0.1436</td>
<td>-0.1428</td>
<td>-0.1437</td>
<td>-0.1456</td>
</tr>
<tr>
<td>-0.1395</td>
<td>0.4277</td>
<td>-0.1468</td>
<td>1.0000</td>
<td>-0.1381</td>
<td>-0.1436</td>
<td>-0.1428</td>
<td>-0.1437</td>
<td>-0.1456</td>
<td>-0.1437</td>
</tr>
<tr>
<td>-0.1428</td>
<td>-0.1455</td>
<td>-0.1437</td>
<td>-0.1381</td>
<td>1.0000</td>
<td>-0.1456</td>
<td>-0.1394</td>
<td>-0.1468</td>
<td>-0.1437</td>
<td>-0.1437</td>
</tr>
<tr>
<td>-0.1467</td>
<td>0.4286</td>
<td>-0.1395</td>
<td>-0.1436</td>
<td>-0.1456</td>
<td>1.0000</td>
<td>-0.1437</td>
<td>-0.1428</td>
<td>-0.1380</td>
<td>-0.1437</td>
</tr>
<tr>
<td>-0.1380</td>
<td>-0.7151</td>
<td>0.1456</td>
<td>-0.1428</td>
<td>-0.1394</td>
<td>1.0000</td>
<td>-0.1437</td>
<td>-0.1468</td>
<td>-0.1437</td>
<td>-0.1437</td>
</tr>
<tr>
<td>-0.1456</td>
<td>0.4286</td>
<td>-0.1380</td>
<td>-0.1437</td>
<td>-0.1468</td>
<td>-0.1428</td>
<td>-0.1437</td>
<td>1.0000</td>
<td>-0.1394</td>
<td>-0.1437</td>
</tr>
<tr>
<td>-0.1437</td>
<td>-0.1381</td>
<td>-0.1428</td>
<td>-0.1456</td>
<td>-0.1437</td>
<td>-0.1380</td>
<td>-0.1468</td>
<td>-0.1394</td>
<td>1.0000</td>
<td>-0.1437</td>
</tr>
</tbody>
</table>

### Table 4.3-6: Cross Correlation Coefficients of the 9 Output Signals Using QPSK With the $W(t)$ Function

<table>
<thead>
<tr>
<th></th>
<th>1.0000</th>
<th>-0.1431</th>
<th>-0.1411</th>
<th>-0.1451</th>
<th>-0.1431</th>
<th>-0.1411</th>
<th>-0.1431</th>
<th>-0.1431</th>
<th>-0.1431</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>-0.1431</td>
<td>1.0000</td>
<td>-0.1451</td>
<td>0.4294</td>
<td>-0.1411</td>
<td>0.4294</td>
<td>-0.7137</td>
<td>0.4274</td>
<td>-0.1451</td>
</tr>
<tr>
<td>-0.1431</td>
<td>1.0000</td>
<td>-0.1451</td>
<td>1.0000</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>-0.1451</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
<tr>
<td>-0.1411</td>
<td>-0.1451</td>
<td>1.0000</td>
<td>-0.1411</td>
<td>1.0000</td>
<td>-0.1431</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
<tr>
<td>-0.1451</td>
<td>0.4294</td>
<td>-0.1411</td>
<td>1.0000</td>
<td>-0.1431</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
<tr>
<td>-0.1431</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>1.0000</td>
<td>-0.1431</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
<tr>
<td>-0.1411</td>
<td>-0.1451</td>
<td>-0.1451</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>1.0000</td>
<td>-0.1431</td>
<td>-0.1411</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
<tr>
<td>-0.1431</td>
<td>0.4274</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1411</td>
<td>-0.1411</td>
<td>1.0000</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
<tr>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>-0.1431</td>
<td>1.0000</td>
<td>-0.1431</td>
<td>-0.1431</td>
</tr>
</tbody>
</table>

### Table 4.3-7: Cross Correlation Coefficients of the 9 Output Signals Using QPSK Without the $W(t)$ Function

<table>
<thead>
<tr>
<th></th>
<th>1.0000</th>
<th>-0.1475</th>
<th>-0.1675</th>
<th>-0.0918</th>
<th>-0.1667</th>
<th>-0.0918</th>
<th>-0.0908</th>
<th>-0.3541</th>
<th>-0.1667</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>-0.1675</td>
<td>1.0000</td>
<td>-0.1658</td>
<td>0.5479</td>
<td>-0.1667</td>
<td>0.5479</td>
<td>-0.7302</td>
<td>0.4713</td>
<td>-0.1658</td>
</tr>
<tr>
<td>-0.1658</td>
<td>1.0000</td>
<td>-0.1675</td>
<td>1.0000</td>
<td>-0.0909</td>
<td>-0.1670</td>
<td>-0.0909</td>
<td>-0.3533</td>
<td>-0.1670</td>
<td>-0.1658</td>
</tr>
<tr>
<td>-0.0918</td>
<td>0.5479</td>
<td>-0.0909</td>
<td>1.0000</td>
<td>-0.0902</td>
<td>-0.3996</td>
<td>-0.4001</td>
<td>0.2582</td>
<td>-0.0925</td>
<td>-0.3537</td>
</tr>
<tr>
<td>-0.1667</td>
<td>-0.1670</td>
<td>-0.0902</td>
<td>1.0000</td>
<td>-0.0925</td>
<td>-0.0914</td>
<td>-0.3537</td>
<td>-0.1658</td>
<td>-0.1670</td>
<td>-0.0925</td>
</tr>
<tr>
<td>-0.0918</td>
<td>0.5479</td>
<td>-0.0909</td>
<td>0.3996</td>
<td>-0.0925</td>
<td>1.0000</td>
<td>-0.4001</td>
<td>0.2582</td>
<td>-0.0902</td>
<td>-0.3537</td>
</tr>
<tr>
<td>-0.0908</td>
<td>-0.7302</td>
<td>-0.0921</td>
<td>-0.4001</td>
<td>-0.0914</td>
<td>-0.4001</td>
<td>1.0000</td>
<td>-0.2585</td>
<td>-0.0914</td>
<td>-0.3537</td>
</tr>
<tr>
<td>-0.3541</td>
<td>0.4713</td>
<td>-0.3533</td>
<td>0.2582</td>
<td>-0.3537</td>
<td>0.2582</td>
<td>0.2585</td>
<td>1.0000</td>
<td>-0.3537</td>
<td>-0.3537</td>
</tr>
<tr>
<td>-0.1667</td>
<td>-0.1670</td>
<td>-0.0925</td>
<td>-0.1658</td>
<td>-0.0902</td>
<td>-0.0914</td>
<td>-0.3537</td>
<td>1.0000</td>
<td>-0.0925</td>
<td>-0.1658</td>
</tr>
</tbody>
</table>
The tables of coefficient values were constructed using a simple program written in MATLAB which is shown below.

```matlab
for i=1:9
    for j=1:9
        corr = corrcoeff(matrix_a(i,:),matrix_b(j,:));
        coeff(i,j) = corr(1,2);
    end
end
```

Where `matrix_a` is a matrix consisting of the matrices of the stored variables (either the PN sequences or the transmitter output signals). Each matrix of the variables is stored on a different row. Hence there will be nine rows corresponding to the nine users. A copy of `matrix_a` is made and stored in `matrix_b`, then the program is executed. The variable `corr` is a 2x2 matrix consisting of the cross correlation coefficients of:

1. Signal 1 with itself (row one, column one)
2. Signal 1 with signal 2 (row one, column two)
3. Signal 2 with signal 1 (row two, column one)
4. Signal 2 with itself (row two, column two)

We are interested in either the value of row one, column two or row two, column one. These two values represent the same correlation coefficient between the two signals. Hence the program stores the value in row one, column two in variable `coeff` which is a 9x9 matrix.

It can be seen that the simulation values for the PN sequence correlation coefficients are quite accurate. These values nearly match the actual PN sequence coefficients given in Table 4.3-1. The cross correlation coefficients for the output signals are also quite reasonable, although they do not match the PN sequence correlations exactly.
4.4 Cross Correlation Values With Delay

The cross correlation coefficients were found between selected signals where one of the signals was delayed by a certain number of chips. For these simulations, the output signals from user 1, user 3 and user 7 were randomly chosen as the signals to be delayed. The cross correlation coefficients were found for a delay varying from 1 to 14 chips. This was done for both BPSK and QPSK modulations with and without the $W(t)$ function.

All simulations for BPSK modulation were done over a period of four simulation seconds. Since the $W(t)$ function has a period four times the length of the PN sequence, there is one $W(t)$ function for four PN sequences. Thus the simulations were run for four seconds so that the $W(t)$ function would have an effect on the cross correlation coefficients. The matrix holding the data had a maximum of 2800 timesteps. The simulation step size was $1/700$ so that in four simulation seconds, a matrix of 2800 timestep values was generated. Exactly four BPSK symbols and four PN sequences are generated in four simulation seconds. Since 1 chip is $1/7$ seconds, a delay of 1 chip is $700/7 = 100$ shifts of the elements in the output signal matrix. A delay of 5 chips corresponds to a matrix shift of $700*5/7 = 500$. A simple program was written in MATLAB to delay the signal by a certain number of chips, find the cross correlation coefficient between the delayed signal and another signal, and then plot this correlation coefficient value. The program was repeated so the coefficients for delays from 0 chips to 14 chips were plotted. This program is shown below:

```matlab
sig = bout1;

for delay = 0:1400
    delayed_sig = [sig((2800-delay+1):2800), sig(1:2800-delay)];
    corr = corrcoeff(delayed_sig, bout5);
    coeff_mat(delay+1) = corr(1,2);
end
plot(coeff_mat);
```
This program is for the example of BPSK modulation. The variable \( \text{sig} \) is the output signal to be delayed. The transpose of the output signal is found so that it will become a row matrix and hence can be treated as a vector. The program is repeated from a delay of 0 chips to 14 chips. There are 100 timesteps per chip hence the delay is increased from 0 to 1400. The variable \( \text{delay} \) corresponds to how many elements the vector must be shifted. The signal is shifted and stored in \( \text{delayed}_\text{sig} \). The cross correlation coefficient is then found between the delayed signal and another output signal. In this case, the output signal from user 5 is used. The transpose of \( \text{delayed}_\text{sig} \) is found to convert the vector back to a column matrix. The element in the first row and second column of the correlation coefficient matrix is stored in a vector \( \text{coeff}_\text{mat} \) because this is the actual correlation coefficient value between the two signals. This vector has 700 elements when the program stops running. The vector of coefficient values is then plotted.

For QPSK modulation, the simulations were done over a period of eight simulation seconds. Hence a matrix of 5600 timesteps was generated since the simulation step size was 1/700. In eight simulation seconds, exactly four QPSK symbols and four PN sequences are generated. For QPSK, one chip is 2/7 seconds. Hence a delay of 1 chip is \( 2/7 \times 700 = 200 \) shifts of the elements in the output signal matrix. For a delay of 5 chips, the matrix elements must be shifted by \( 10/7 \times 700 = 1000 \). The same program was used to plot the cross correlation coefficient values except that the value of 5600 was used instead of 2800 and the delay was varied from 0 to 2800 which corresponds to a delay of 0 to 14 chips. The program is shown below:

```matlab
sig = qout7';
for delay = 0:2800
    delayed_sig = [sig((5600-delay+1):5600),sig(1:5600-delay)];
    corr = corrcoef(delayed_sig',qout6);
    coeff_mat(delay+1) = corr(1,2);
end
plot(coeff_mat);
```
In this case the delayed signal is the output signal from user 7 and the cross correlation coefficients are found between this delayed signal and the output signal from user 6.

4.4.1 The Cross Correlation Coefficients Between Signals Using BPSK Modulation With and Without the $W(t)$ Function.

The cross correlation coefficients between delayed signals and output signals using BPSK and the $W(t)$ function are graphically shown in Figures 4.4.1-1a, 4.4.1-2a and 4.4.1-3a. The output signals from user 1, user 3 and user 7 were chosen to be delayed. For the cross correlation coefficients between the signals without the $W(t)$ function, refer to Figures 4.4.1-1b, 4.4.1-2b and 4.4.1-3b.

4.4.2 The Cross Correlation Coefficients Between Signals Using QPSK Modulation With and Without the $W(t)$ Function.

The cross correlation coefficients between delayed signals and output signals using QPSK and the $W(t)$ function are graphically shown in Figures 4.4.2-1a, 4.4.2-2a and 4.4.2-3a. The same delayed signals as for BPSK were used. For the cross correlation coefficients between the signals without the $W(t)$ function, refer to Figures 4.4.2-1b, 4.4.2-2b and 4.4.2-3b.
Figure 4.4.1-1a Cross Correlation Coefficients Between the Output Signals of Users 1 and 4 Using BPSK With the $W(t)$ Function.

Figure 4.4.1-1b Cross Correlation Coefficients Between the Output Signals of Users 1 and 4 Using BPSK Without the $W(t)$ Function.
SJMULINK Implementation of a CDMA Transmitter

Figure 4.4.1-2a Cross Correlation Coefficients Between the Output Signals of Users 3 and 5 Using BPSK With the $W(t)$ Function.

Figure 4.4.1-2b Cross Correlation Coefficients Between the Output Signals of Users 3 and 5 Using BPSK Without the $W(t)$ Function.
SIMULINK Implementation of a CDMA Transmitter

Figure 4.4.1-3a Cross Correlation Coefficients Between the Output Signals of Users 7 and 6 Using BPSK With the \( W(t) \) Function.

Figure 4.4.1-3b Cross Correlation Coefficients Between the Output Signals of Users 7 and 6 Using BPSK Without the \( W(t) \) Function.
Figure 4.4.2-la Cross Correlation Coefficients Between the Output Signals of Users 1 and 4 Using QPSK With the $W(t)$ Function.

Figure 4.4.2-lb Cross Correlation Coefficients Between the Output Signals of Users 1 and 4 Using QPSK Without the $W(t)$ Function.
Figure 4.4.2-2a Cross Correlation Coefficients Between the Output Signals of Users 3 and 5 Using QPSK With the $W(t)$ Function.

Figure 4.4.2-2b Cross Correlation Coefficients Between the Output Signals of Users 3 and 5 Using QPSK Without the $W(t)$ Function.
Figure 4.4.2-3a Cross Correlation Coefficients Between the Output Signals of Users 7 and 6 Using QPSK With the \( W(t) \) Function.

Figure 4.4.2-3b Cross Correlation Coefficients Between the Output Signals of Users 7 and 6 Using QPSK Without the \( W(t) \) Function.
4.4.3 Analysis

It can be seen that the inclusion of the $W(t)$ function in the modulation greatly improves the cross correlation coefficients of two signals when one of them experiences a delay. It can be observed that after a delay of 5 chips, the cross correlation coefficients generally diminish in value and are significantly less than without the modified carrier function $W(t)$. The cross correlation coefficients repeat themselves after a delay of 7 chips without the $W(t)$ function hence the values are higher than with the function.

The introduction of frequency modulation with the modulating function has the effect of making the magnitude of the autocorrelation function of the carrier waveform lower than unity for all possible delays $T_e < \tau \leq 25T_e$ [3]. This is the reason for the reduced cross correlation coefficient values.

The values of $\alpha$ and $\beta$ used in the simulations were 3.72 and 0.2 respectively. These values are only arbitrary and the cross correlation coefficients may be optimized by varying these values.

4.5 Average Power Spectral Density of the Output Signals

The average power spectral density (PSD) of the 9 output signals for both BPSK modulation and QPSK modulation was plotted. The modulation included the $W(t)$ function. For BPSK modulation, the carrier frequency was 28Hz and for QPSK, the carrier frequency was 14Hz. Hence the carrier is four times the chip rate in both cases. The power spectral density was plotted using MATLAB's 'PSD' function which plots the power spectral density of the waveform for the positive frequencies.

4.5.1 Average PSD of the Output Signals Using BPSK Modulation

In BPSK modulation, the average power spectral density was taken for 64 samples of the data. This meant that the number of transmitted bits was
The command to plot the average power spectral density was

\[ \text{PSD}(\text{bout}, [], 80) \]

where 'bout' is the variable name which stores the output signal and 80 is the sampling frequency. This value was obtained from the reciprocal of the simulation step size \((1/0.0125)\). The empty matrix indicates that the default value of the NFFT was used. This value is 256. All other parameters were default values. The PSD plots for BPSK are shown in Figures 4.5.1-1 to 4.5.1-9.

Notice that the highest power magnitude is for the carrier frequency of 28Hz. It can be observed that the peaks diminish in amplitude as the distance from the carrier increases. This distance between the peaks is constant at 7Hz, which corresponds to the chip rate. The reason that the amplitudes are different for each simulation is that each PN sequence will have its own spectral properties. This is then reflected in the spectral properties of the output signal.
SIMULINK Implementation of a CDMA Transmitter

Figure 4.5.1-1 Simulation 1

Figure 4.5.1-2 Simulation 2

Figure 4.5.1-3 Simulation 3
SIMULINK Implementation of a CDMA Transmitter

Figure 4.5.1-4 Simulation 4

Figure 4.5.1-5 Simulation 5

Figure 4.5.1-6 Simulation 6
Figure 4.5.1-7 Simulation 7

Figure 4.5.1-8 Simulation 8

Figure 4.5.1-9 Simulation 9
4.5.2 Average PSD of the Output Signals Using QPSK Modulation

In QPSK modulation, the average PSD was also taken for 64 samples of the QPSK symbols. The simulation time was 2048 seconds because one PN sequence is generated in 2 seconds for QPSK. The simulation step size was the same as for BPSK (0.0125). The matrix containing the output signal had a maximum size of 163840 timesteps. The PSD was also plotted in the same way as for BPSK with the same parameters.

The average PSD plots for QPSK are shown in Figures 4.5.2-1 to 4.5.2-9.

Notice that the highest power magnitude is for the carrier frequency of 14Hz. The distance between the peaks of the PSD is around 3.5Hz. This frequency corresponds to the chip rate for QPSK. The amplitudes for each simulation are different due to the fact that the PN sequences have their own spectral properties. Thus the output signals have different amplitudes.
SIMULINK Implementation of a CDMA Transmitter

Figure 4.5.2-1 Simulation 1

Figure 4.5.2-2 Simulation 2

Figure 4.5.2-3 Simulation 3
SIMULINK Implementation of a CDMA Transmitter

**Figure 4.5.2-4 Simulation 4**

![Power Spectrum Magnitude (dB) vs Frequency Graph](image)

**Figure 4.5.2-5 Simulation 5**

![Power Spectrum Magnitude (dB) vs Frequency Graph](image)

**Figure 4.5.2-6 Simulation 6**

![Power Spectrum Magnitude (dB) vs Frequency Graph](image)
SIMULINK implementation of a CDMA Transmitter

Figure 4.5.2-7 Simulation 7

Figure 4.5.2-8 Simulation 8

Figure 4.5.2-9 Simulation 9
5 Further Development: Digital Signal Processing Implementation

A digital signal processing (DSP) implementation of a CDMA transmitter could be a further development of the SIMULINK implementation. The benefits of using DSP will be discussed in section 5.1. A description of the current technology in DSP which would be suitable for the hardware implementation is given in section 5.2.

5.1 Benefits of Using DSP

Digital signal processing (DSP) is interested in the digital representation of signals and the use of digital processors to analyse, modify, or extract information from the signals [22]. The signals used in most kinds of DSP are analogue signals which have been sampled at regular intervals and converted to a digital form.

The reason for processing a digital signal may be, for example to remove interference or noise from the signal. Other reasons may be to obtain the spectrum of the data or to transform the signal into a more suitable form. The main advantages of DSP are [22]:

- **Secured accuracy.** The number of bits used is the only determining factor in accuracy.
- **Excellent reproducibility.** There are no variations due to component tolerances hence identical performance from unit to unit is obtained.
- There is no variation in performance with temperature and age.
- The advances in semiconductor technology to achieve greater reliability, smaller size, higher speed, low power consumption and lower cost have been tremendous. Thus advantage of this advance in technology is always taken. For example, it is now possible to produce high speed low power integrated circuits (ICs) using CMOS technology. This has resulted in newer DSP chips being CMOS devices rather than bipolar.
Better flexibility. One of the most important features of DSP is that it allows systems to be programmed and reprogrammed to perform different functions without modifying the hardware.

Superior performance. The functions that are not possible to perform with analogue signal processing can be done using DSP. Some examples of what DSP can achieve are linear phase response and the implementation of complex adaptive filtering algorithms.

DSP may be the only feasible option in some cases because the information may already be in a digital form.

There are some disadvantages associated with DSP but new technology is continually diminishing the significance of these disadvantages.

Cost and speed. When large bandwidth signals are being used in DSP designs, it can be expensive. Most DSP devices are still not fast enough and can only process signals of moderate bandwidth. However, DSP devices being developed currently are becoming faster and faster.

Design time. DSP designs can be time consuming and almost impossible in some cases unless you have the necessary resources and are knowledgeable in DSP techniques. This situation is changing though, and commercial companies are starting to use the advantages of DSP in their products. Also, new graduate engineers now possess some knowledge of digital techniques.

Problems with finite wordlengths. Financial considerations in real-time situations usually mean that only a limited number of bits can be used to implement DSP algorithms. Sometimes, an inadequate number of bits is used to represent variables and this may result in serious degradation in system performance.
5.2 Hardware Description

The DSP board that would be suitable for the implementation is the TMS320C8x Software Development Board. "The Software Development Board (SDB) is a PC/AT plug-in card that allows you to evaluate characteristics of the TMS320C8x DSP to determine how it will meet the requirements of your given application. Software engineers, developers, and programmers can also use the SDB as a development tool to create software and applications on a PC for the TMS320C8x" [23].

Some of the features of the SDB are [23]:

- 40 MHz TMS320C80
- 16-bit 44.1-kHz stereo audio subsystem
- 16-bit video acquisition subsystem
- 8 Mbytes DRAM (program memory)
- 4 Mbytes display VRAM to support 16-bit 1024x768 video applications
- SDB Source-Level Debugger
- PCI bus master interface
- 'C8x programming examples
- On-board emulation support

The chip used on the board is the Texas Instruments TMS320C80. It is currently one of the fastest processors available. It is a single chip, parallel processor that can be used for applications such as image processing and audio/video digital compression. The processing power of the 'C80 also supports applications within the digital telecom, security and image recognition markets [24].

The TMS320C80 integrates onto a single IC, five powerful fully programmable processors, a sophisticated DMA (direct memory access) controller with a DRAM, SRAM, and VRAM external memory interface, 50K bytes of SRAM and video timing control. Of the 50K bytes of SRAM, 32K bytes are shared among the five processors to support various parallel processing approaches. Applications that require a large amount of processing and a wide range of
multimedia can be facilitated by this unique combination of hardware. All five processors can be programmed in both C and assembly language.

The 'C80 is capable of performing over two billion RISC like operations per second. In some applications, a single 'C80 can do the job of over ten of the most powerful DSPs or general-purpose processors previously available. During each second of processing, it can move 1.8 Gbytes of instructions within the chip, 2.4 Gbytes of data and 400 Mbytes of data to off-chip memory.

The fully scannable design of the 'C80 combined with its flexible architecture and in-circuit emulation capability, lets you design a system that meets your specific needs and can replace multiple boards and multiple processors. Of the many benefits of the TMS320C80, the most relevant to this project is that the chip gives you the processing power to support telecommunication applications such as cellular base stations and telephone networks. Since these applications require a great amount of processing power, the 'C80 is ideal for these purposes. A typical cellular base station today uses over 4000 DSPs in a single station. The TMS320C80 can reduce this number considerably. The 'C80 also offers high data bandwidth and interprocessor communication. This is done through a combination of many small, 64-bit wide RAMs that are interconnected. An intelligent DMA controller, the transfer controller, handles the movement of blocks of data [24]. This makes sure that the processors do not have to wait on data and that the interprocessor communication does not bottleneck.

This DSP implementation can be carried out quite easily by converting the SIMULINK code to C code and downloading the algorithms onto the DSP board. Similar testing done for the SIMULINK model may then be conducted using the DSP tools.
6 Conclusions

The aim of this project was to develop a SIMULINK implementation of a CDMA transmitter. The transmitter was to reduce intersymbol interference and multiple access interference using a non-conventional modulating function (the $W(t)$ function). This objective has been achieved with the simulation results indicating that the transmitter is performing correctly.

From the average PSD of the output signals, it has been shown that the highest magnitude of the PSD has a corresponding frequency component which is the carrier frequency. This is true for both BPSK and QPSK modulation. For BPSK, the distance between the peaks of the PSD is 7Hz, which corresponds to the chip rate. For QPSK, the chip rate is half that of BPSK, thus the distance between the peaks is 3.5Hz. The PN sequences have different spectral properties and this was reflected in the PSD of the output signals.

The most significant outcome of this project is that the use of the $W(t)$ function in the modulation of the data seems to improve the cross correlation coefficients between the transmitter output signals when there is a delay. It was found that generally there was a gradual reduction in the cross correlation coefficients after delays of 5 chips. The carrier waveform was modified such that the normalized to the energy per bit autocorrelation function of the carrier wave is a periodic function of the delay. The period is greater than the maximum relative delay between any non-negligible multipath signals or between any two channels. Also, the normalized to the energy per bit autocorrelation function magnitude is always less than unity apart from the case when its argument is equal to an integer multiple of the autocorrelation period [3]. Thus this modification results in significantly lower cross correlation coefficient values than without the inclusion of the $W(t)$ function. In a practical situation, such as the urban environment where reflections from buildings are expected, multipath propagation arises and consequently delays in the received signals occur. This scheme is of benefit in those circumstances.
Further development of the SIMULINK implementation is a DSP implementation of a CDMA transmitter. The reason for developing a DSP implementation of the transmitter is mainly due to the greater flexibility and superior performance that DSP provides. The DSP board that was suggested to be used in the implementation has the processing power to support telecommunication applications. It also offers high data bandwidth and interprocessor communication. These factors make DSP technology ideal for the purposes of a practical implementation of the CDMA transmitter presented in this thesis.
SIMULINK Implementation of a CDMA Transmitter

References


Appendix A: SIMULINK Implementation of a CDMA Transmitter Using BPSK

slm5h.m

function [ret,x0,str,ts,xts]=simSh(t,x,u,flag);
%SIM5H is the M-file description of the SIMULINK system named SIM5H.
% The block-diagram can be displayed by typing: SIM5H.
% SYS=SIM5H(T,X,U,FLAG) returns depending on FLAG certain
% system values given time point, T, current state vector, X,
% and input vector, U.
% FLAG is used to indicate the type of output to be returned in SYS.
% Setting FLAG=1 causes SIM5H to return state derivatives, FLAG=2
% discrete states, FLAG=3 system outputs and FLAG=4 next sample
% time. For more information and other options see SFUNC.
% Calling SIM5H with a FLAG of zero:
% SIZES=SIM5H([],[],[],0,) returns a vector, SIZES, which
% contains the sizes of the state vector and other parameters.
% SIZES(1) number of states
% SIZES(2) number of discrete states
% SIZES(3) number of outputs
% SIZES(4) number of inputs
% SIZES(5) number of roots (currently unsupported)
% SIZES(6) direct feedthrough flag
% SIZES(7) number of sample times
% For the definition of other parameters in SIZES, see SFUNC.
% See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEARG.

% Note: This M-file is only used for saving graphical information;
% after the model is loaded into memory an internal model
% representation is used.

% the system will take on the name of this mfile:
sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
    set_param(sys,'Location',[4,42,788,588])
    open_system(sys)
end;
set_param(sys,'algorithm', 'RK-45')
set_param(sys,'Start time', '0.0')
set_param(sys,'Stop time', '4')
set_param(sys,'Min step size', '0.001428571429')
set_param(sys,'Max step size', '0.001428571429')
set_param(sys,'Relative error','1e-6')
set_param(sys,'Return vars', '')

add_block('built-in/Fcn',[sys,'/','Fen'])
set_param([sys,'/','Fcn'],
    'Expr','rem(u,16)',
    'position',[170,75,210,95])

add_block('built-in/Scope',[sys,'/','Data'])
set_param([sys,'/','Data'],
    'Vgain',2.000000',
    'Hgain',8.000000',
    'Vmax',4.000000',
    'Hmax',16.000000',
    'Window',[6,54,786,289],
    'position',[420,17,440,43])

add_block('built-in/MATLAB Fcn',[sys,'/','Fix'])
set_param([sys,'/','Fix'],
    'MATLAB Fcn','fix',
    'position',[250,70,300,100])

add_block('built-in/Fcn',[sys,'/','Fen1'])
set_param([sys,'/','Fcn1'],
    'Expr','rem(u,7)',
    'position',[190,400,230,420])

add_block('built-in/Scope',[sys,'/','Scope4'])
set_param([sys,'/','Scope4'],
    'Vgain',5.000000',
    'Hgain',5.000000',
    'Vmax',10.000000',
    'Hmax',10.000000',
    'Window',[2,41,324,263],
    'position',[260,477,280,503])

add_block('built-in/MATLAB Fcn',[sys,'/','Fix'])
set_param([sys,'/','Fix'],
    'MATLAB Fcn','fix',
    'position',[270,395,320,425])

% Subsystem 'PN Sequence'.
new_system([sys,'/','PN Sequence'])
set_param([sys,'/','PN Sequence'],[316,263,598,417])
SIMULINK Implementation of a CDMA Transmitter

add_block('built-in/Inport',[sys,'/','PN Sequence/y0'])
set_param([sys,'/','PN Sequence/y0'],...
    'Port',2,...
    'position',[20,80,40,100])

add_block('built-in/Outport',[sys,'/','PN Sequence/table out'])
set_param([sys,'/','PN Sequence/table out'],...
    'position',[215,55,235,75])

add_block('built-in/Inport',[sys,'/','PN Sequence/x0'])
set_param([sys,'/','PN Sequence/x0'],...
    'position',[20,25,40,45])

add_block('built-in/S-Function',[sys,'/','PN Sequence/S-function'])
set_param([sys,'/','PN Sequence/S-function'],...
    'function name','sftable2',...
    'parameters',xindex, yindex, table',...
    'position',[140,52,190,78])

add_block('built-in/Mux',[sys,'/','PN Sequence/Mux'])
set_param([sys,'/','PN Sequence/Mux'],...
    'inputs',2,...
    'position',[90,46,120,79])

add_line([sys,'/','PN Sequence'],[45,90;65,90;65,70;85,70])
add_line([sys,'/','PN Sequence'],[195,65;210,65])
add_line([sys,'/','PN Sequence'],[45,35;65,35;65,55;85,55])
add_line([sys,'/','PN Sequence'],[125,65;135,65])
set_param([sys,'/','PN Sequence'],...
    'Mask Display',plot(-10,-10,110,110,[90,50,10],[90,40,30],[90,50,10],[50,26,20],[90,50,10],[22,13,10])',...
    'Mask Type','2-D Table Lookup')

set_param([sys,'/','PN Sequence'],...
    'Mask Dialogue','Two Dimensional Table Lookup. The first input corresponds to X Index and the second input corresponds to the Y Index(X Index|Y Index|Table)')

set_param([sys,'/','PN Sequence'],...
    'Mask Translate',xindex=@1; yindex=@2; table=@3;
sftab2chk(xindex,yindex,table);')

set_param([sys,'/','PN Sequence'],...
    'Mask Help','This block returns a linearly interpolated intersection from the table using the X index (which corresponds to the rows of the table) and the Y index (which corresponds to the columns of the table). Extrapolation is used.')

set_param([sys,'/','PN Sequence'],...
    'Mask Entries','[ 1 2 3 4 5 6 7 8 9 ]V[0 1 2 3 4 5 6 ]V[1 -1 1 1 1 -1 -1; 1 1 -1 -1 -1 1 -1; 1 1 -1 -1 -1 1 -1; 1 -1 -1 -1 1 1 1; -1 -1 -1 1 1 1 -1; -1 -1 -1 -1 1 1 1 ; -1 -1 -1 -1 -1 1 1 ]V')

% Finished composite block 'PN Sequence'.

97
SIMULINK Implementation of a CDMA Transmitter

set_param([sys,'/','PN Sequence'],...
    'position',[370,379,415,421])

add_block('built-in/To Workspace',[sys,'/','To Workspace2'])
set_param([sys,'/','To Workspace2'],...
    'mat-name','pnout6',...
    'position',[455,312,505,328])

add_block('built-in/To Workspace',[sys,'/','To Workspace1'])
set_param([sys,'/','To Workspace1'],...
    'mat-name','pout',...
    'position',[625,32,675,48])

add_block('built-in/Scope',[sys,'/','Output'])
set_param([sys,'/','Output'],...
    'Vgain','2.120000',...
    'Hgain','2.120000',...
    'Vmax','4.240000',...
    'Hmax','4.240000',...
    'Window',[42,101,728,310],...
    'position',[720,282,740,308])

add_block('built-in/Gain',[sys,'/','cos(Wct+integral[W(t)dt])/W(t)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/W(t)'],...
    'Gain','2/9*3.72',...
    'position',[105,155,125,175])

add_block('built-in/Gain',[sys,'/','cos(Wct+integral[W(t)dt])/2//9*alpha*beta'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/2//9*alpha*beta'],...
    'Gain','2/9*3.72*0.3',...
    'position',[105,155,125,175])

new_system([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t//14Tc)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t//14Tc)'],'Location',[4,104,375,363])
add_block('built-in/Gain', [sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Gain1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Gain1'],
'Gain','pi/2',
'position',[100,55,120,75])

add_block('built-in/Clock',[sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Clock'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Clock'],
'position',[60,55,80,75])

add_block('built-in/Sum', [sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Tr(t)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Tr(t)'],
'inputs','+',
'position',[230,150,250,170])

add_block('built-in/Constant', [sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Constant'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Constant'],
'position',[185,175,205,195])

add_block('built-in/MATLAB Fcn', [sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Sawtooth'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Sawtooth'],
'MATLAB Fcn','sawtooth',
'position',[65,125,115,155])

add_block('built-in/Scope', [sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Scope'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/Scope'],
'Vgain','1.000000',
'Hgain','2.000000',
'Vmax','4.000000',
'Hmax','4.000000',
'Window',[297,108,619,488],
'position',[295,82,315,108])

add_block('built-in/Outport', [sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/out_1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)/out_1'],
'position',[270,150,290,170])

add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)'],[210,185;215,185;225,165])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/14Tc)'],[210,140;215,140;225,155])
SIMULINK Implementation of a CDMA Transmitter

```matlab
add_line([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'],[125,65;145,100;45,100;45,140;60,140])
add_line([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'],[85,65;95,65])
add_line([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'],[170,140;180,140])
add_line([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'],[120,140;135,140])
add_line([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'],[255,160;265,160])
add_line([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'],[255,160;255,95;290,95])

% Finished composite block 'cos(Wct+integral[W(t)dt])f(t/14Tc)'.
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/14Tc)'], 'position', [35,140,65,190])

% Subsystem 'cos(Wct+integral[W(t)dt])f(t/77Tc)'.
new_system([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)'], 'Location', [4,104,594,363])
add_block('built-in/Gain', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Gain'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Gain'], 'Gain', pi,...
    'position', [100,55,120,75])
add_block('built-in/Clock', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Clock'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Clock'],
    'position', [60,55,80,75])
add_block('built-in/Sum', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Tr(t)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Tr(t)'],
    'Inputs', '+-', ...'position', [230,150,250,170])
add_block('built-in/Constant', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Constant'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Constant'],
    'position', [185,175,205,195])
add_block('built-in/Gain', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Gain'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Gain'],
    'Gain', '2', ...
    'position', [185,130,205,150])
add_block('built-in/Abs', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Abs'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Abs'],
    'position', [140,130,165,150])
add_block('built-in/MATLAB Fcn', [sys,'/','cos(Wct+integral[W(t)dt])f(t/77Tc)/Sawtooth'])
```
set_param(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/Sawtooth'),...
    'MATLAB Fcn','sawtooth',...
    'position',[65,125,115,155])

add_block('built-in/Scope',sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/Scope'])
set_param(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/Scope'],...
    'Vgain','1.000000',...
    'Hgain','2.000000',...
    'Vmax','2.000000',...
    'Hmax','4.000000',...
    'Window',[163,220,499,600],...
    'position',[295,82,315,108])

add_line(sys,'/','cos(Wct+integral[W(t)dt])/out_1']
set_param(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'],...
    'position',[270,150,290,170])
add_line(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'],[85,65,95,65])
add_line(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'],[170,140;180,140])
add_line(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'],[255,160;265,160])
add_line(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'],[255,160;255,95;290,95])

% Finished composite block 'cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'.

set_param(sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'),...
    'position',[30,55,60,105])

add_block('built-in/Scope',sys,'/','cos(Wct+integral[W(t)dt])/Scope2'])
set_param(sys,'/','cos(Wct+integral[W(t)dt])/Scope2'],...
    'Vgain','2.000000',...
    'Hgain','1.000000',...
    'Vmax','4.000000',...
    'Hmax','2.000000',...
    'Window',[8,294,773,570],...
    'position',[615,6,635,34])

add_block('built-in/To Workspace',sys,'/','cos(Wct+integral[W(t)dt])/To Workspace 1'])
set_param(sys,'/','cos(Wct+integral[W(t)dt])/To Workspace 1'],...
    'mat-name','yout2',...
    'buffer','2800',...
    'position',[615,107,665,123])

add_block('built-in/Outport',sys,'/','cos(Wct+integral[W(t)dt])/out_1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt]/out_1')],...
'position',[615,50,635,70])

add_block('built-in/MATLAB Fcn',[sys,'/','cos(Wct+integral[W(t)dt]/Cos')])
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Cos'],...
'MATLAB Fcn','cos',...
'position',[535,45,585,75])

% Subsystem 'cos(Wct+integral[W(t)dt]/Wct'.

new_system([sys,'/','cos(Wct+integral[W(t)dt]/Wct']
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct'),'Location',[345,138,634,456])

add_block('built-in/Product',[sys,'/','cos(Wct+integral[W(t)dt]/Wct/Product2')]
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Product2'],...
'position',[105,65,130,85])

add_block('built-in/Clock',[sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Clock')]
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Clock'),...
'position',[45,60,65,80])

add_block('built-in/Scope',[sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Scope5')]
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Scope5'),...
'Vgain',10.000000',...
'Hgain',1.000000',...
'Vmax',20.000000',...
'Hmax',2.000000',...
'Window',[100,100,422,480],...
'position',[170,22,190,48])

add_block('built-in/Outport',[sys,'/','cos(Wct+integral[W(t)dt)]/Wct/out_1')]
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct/out_1'),...
'position',[175,65,195,85])

add_block('built-in/Constant',[sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Wc')]
set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct/Wc'),...
'Value',2*pi*28',...
'position',[50,107,75,123])
add_line([sys,'/','cos(Wct+integral[W(t)dt)]/Wct'],[80,115;90,115;100,80])
add_line([sys,'/','cos(Wct+integral[W(t)dt)]/Wct'],[70,70;100,70])
add_line([sys,'/','cos(Wct+integral[W(t)dt)]/Wct'],[135,75;170,75])
add_line([sys,'/','cos(Wct+integral[W(t)dt)]/Wct'],[135,75;135,35;165,35])

% Finished composite block 'cos(Wct+integral[W(t)dt)]/Wct'.

set_param([sys,'/','cos(Wct+integral[W(t)dt)]/Wct'),...
'position',[410,5,440,55])
add_block('built-in/Sum', [sys, '/', 'cos(Wct+integral[W(t)dt])/Sum'])
set_param([sys, '/', 'cos(Wct+integral[W(t)dt])/Sum'],
  'position', [475, 60, 495, 80])

add_block('built-in/Scope', [sys, '/', 'cos(Wct+integral[W(t)dt])/Scope3'])
set_param([sys, '/', 'cos(Wct+integral[W(t)dt])/Scope3'],
  'Vgain', 1.000000,
  'Hgain', 28.000000,
  'Vmax', 2.000000,
  'Hmax', 56.000000,
  'Window', [8, 220, 334, 584],
  'position', [280, 171, 300, 199])

add_block('built-in/Integrator', [sys, '/', 'cos(Wct+integral[W(t)dt])/Integrator'])
set_param([sys, '/', 'cos(Wct+integral[W(t)dt])/Integrator'],
  'position', [280, 105, 300, 125])

add_block('built-in/Constant', [sys, '/', 'cos(Wct+integral[W(t)dt])/2*pi'])
set_param([sys, '/', 'cos(Wct+integral[W(t)dt])/2*pi'],
  'Value', 2*pi,
  'position', [280, 60, 300, 80])

add_block('built-in/Product', [sys, '/', 'cos(Wct+integral[W(t)dt])/Product'])
set_param([sys, '/', 'cos(Wct+integral[W(t)dt])/Product'],
  'position', [355, 100, 380, 120])

add_block('built-in/Scope', [sys, '/', 'cos(Wct+integral[W(t)dt])/Scope1'])
set_param([sys, '/', 'cos(Wct+integral[W(t)dt])/Scope1'],
  'Vgain', 0.500000,
  'Hgain', 28.000000,
  'Vmax', 1.000000,
  'Hmax', 56.000000,
  'Window', [8, 296, 771, 584],
  'position', [485, 146, 505, 174])

add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [500, 70; 520, 70; 530, 60])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [445, 50; 460, 30; 470, 65])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [130, 165; 175, 165; 175, 125; 195, 125])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [130, 80; 185, 80; 195, 105])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [65, 80; 100, 80])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [70, 165; 100, 165])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [590, 60; 610, 60])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [590, 60; 590, 115; 610, 115])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [590, 60; 590, 20; 610, 20])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [240, 115; 250, 115; 250, 185; 275, 185])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [250, 115; 275, 115])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [305, 70; 325, 70; 325, 105; 350, 105])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [305, 115; 350, 115])
add_line([sys, '/', 'cos(Wct+integral[W(t)dt])'], [385, 110; 460, 110; 470, 75])
SIMULINK Implementation of a CDMA Transmitter

```matlab
add_line(sys,'/','cos(Wct+integral[W(t)dt]),[425,110;425,160;480,160])

% Finished composite block 'cos(Wct+integral[W(t)dt]).

set_param(sys,'/','cos(Wct+integral[W(t)dt]),...
              'position',[420,125,450,175])

add_block('built-in/Scope',sys,'/','PN')
set_param(sys,'/','PN',...
              'Vgain','2.000000',...
              'Hgain','10.000000',...
              'Vmax','4.000000',...
              'Hmax','20.000000',...
              'Window',[8,354,790,598],...
              'position',[475,425,495,455])

add_block('built-in/Constant',sys,'/','PN Row')
set_param(sys,'/','PN Row',...
              'Value','9',...
              'position',[315,320,335,340])

add_block('built-in/Scope',sys,'/','Scope3')
set_param(sys,'/','Scope3',...
              'Vgain','5.000000',...
              'Hgain','5.000000',...
              'Vmax','10.000000',...
              'Hmax','10.000000',...
              'Window',[49,362,371,574],...
              'position',[345,462,365,488])

add_block('built-in/Scope',sys,'/','Scope1')
set_param(sys,'/','Scope1',...
              'Vgain','20.000000',...
              'Hgain','10.000000',...
              'Vmax','20.000000',...
              'Hmax','20.000000',...
              'Window',[0,16,781,416],...
              'position',[345,132,365,158])

add_block('built-in/Scope',sys,'/','Scope2')
set_param(sys,'/','Scope2',...
              'Vgain','20.000000',...
              'Hgain','16.000000',...
              'Vmax','40.000000',...
              'Hmax','32.000000',...
              'Window',[22,229,686,583],...
              'position',[250,132,270,158])
```
add_block('built-in/Product', [sys,'/','Product1'])
set_param([sys,'/','Product1'], ...
'position',[590,285,615,305])

add_block('built-in/To Workspace', [sys,'/','To Workspace'])
set_param([sys,'/','To Workspace'], ...
'mat-name','bout9', ...
'buffer','2800', ...
'position',[640,187,690,203])

add_block('built-in/Clock', [sys,'/','Clock'])
set_param([sys,'/','Clock'], ...
'position',[25,14,45,36])

add_block('built-in/Clock', [sys,'/','Clock1'])
set_param([sys,'/','Clock1'], ...
'position',[40,135,60,155])

add_block('built-in/Product', [sys,'/','Product2'])
set_param([sys,'/','Product2'], ...
'position',[95,73,125,97])

add_block('built-in/Clock', [sys,'/','Clock2'])
set_param([sys,'/','Clock2'], ...
'position',[55,460,75,480])

add_block('built-in/Product', [sys,'/','Product'])
set_param([sys,'/','Product'], ...
'position',[120,398,150,422])

add_block('built-in/Constant', [sys,'/','Data Rate','13',':'])
set_param([sys,'/','Data Rate','13',':'], ...
'position',[30,70,50,90])

add_block('built-in/Constant', [sys,'/','PN Rate'])
set_param([sys,'/','PN Rate'], ...
'Value','7', ...
'position',[40,395,60,415])

add_block('built-in/Constant', [sys,'/','Constant'])
set_param([sys,'/','Constant'], ...
'position',[280,10,300,30])

% Subsystem 'Data Sequence1'.

new_system([sys,'/','Data Sequence1'])
set_param([sys,'/','Data Sequence1'],'Location',[316,263,598,417])
add_block('built-in/lnport',[sys,'/','Data Sequence I/yO'])
set_param([sys,'/','Data Sequence I/yO'],
    'Port',2,...
    'position',[20,80,40,100])

add_block('built-in/Outport',[sys,'/','Data Sequence I/table out'])
set_param([sys,'/','Data Sequence I/table out'],...
    'position',[215,55,235,75])

add_block('built-in/Inport', [sys, '/','Data Sequence 1/xO'])
set_param([sys,'/','Data Sequence 1/xO'],...
    'position',[20,25,40,45])

add_block('built-in/S-Function', [sys, '/','Data Sequence 1/S-function'])
set_param([sys,'/','Data Sequence 1/S-function'],...
    'function name','sftable2';...
    'parameters','xindex, yindex, table';...
    'position',[140,52,190,78])

add_block('built-in/Mux', [sys, '/','Data Sequence 1/Mux'])
set_param([sys,'/','Data Sequence 1/Mux'],...
    'inputs',2,...
    'position',[90,46,120,79])

add_line([sys,'/','Data Sequence 1'],[45,90;65,90;65,70;85,70])
add_line([sys,'/','Data Sequence 1'],[195,65;210,65])
add_line([sys,'/','Data Sequence 1'],[45,35;65,35;65,55;85,55])
add_line([sys,'/','Data Sequence 1'],[125,65;135,65])
set_param([sys,'/','Data Sequence 1'],...
    'Mask Display',plot(-10,-10,110,110,[90,50,10],[90,40,30],[90,50,10],[50,26,20],[90,50,10],[22,13,10]),...
    'Mask Type','2-D Table Lookup')
set_param([sys,'/','Data Sequence 1'],...
    'Mask Dialogue','Two Dimensional Table Lookup
The first input corresponds to X Index and the second input corresponds to the Y Index IX Index/ Y Index/Table')
set_param([sys,'/','Data Sequence 1'],...
    'Mask Translate','xindex=@1; yindex=@2; table=@3; sftab2chk(xindex,yindex,table);')
set_param([sys,'/','Data Sequence 1'],...
    'Mask Help','This block returns a linearly interpolated intersection from the table using the X index (which corresponds to the rows of the table) and the Y index (which corresponds to the columns of the table). Extrapolation is used.')
set_param([sys,'/','Data Sequence 1'],...
    'Mask Entries',[1 2 3 4 5 6 7 8 9])

106
% Finished composite block 'Data Sequence1'.

set_param([sys, '/', 'Data Sequence1'],...
        'position', [350, 57, 385, 93])

add_block('built-in/Scope', [sys, '/', 'BPSK'])
set_param([sys, '/', 'BPSK'],...
        'Vgain', '1.000000',...
        'Hgain', '10.000000',...
        'Vmax', '2.000000',...
        'Hmax', '20.000000',...
        'Window', [4.0, 769, 244],...
        'position', [665, 68, 685, 92])

add_block('built-in/Product', [sys, '/', 'BPSK_mod'])
set_param([sys, '/', 'BPSK_mod'],...
        'position', [505, 70, 530, 90])

add_line(sys, [130, 85; 165, 85])
add_line(sys, [215, 85; 245, 85])
add_line(sys, [155, 410; 185, 410])
add_line(sys, [235, 410; 265, 410])
add_line(sys, [325, 410; 365, 410])
add_line(sys, [620, 295; 715, 295])
add_line(sys, [390, 75; 405, 75; 415, 30])
add_line(sys, [535, 80; 660, 80])
add_line(sys, [420, 400; 450, 400; 450, 300; 485, 300])
add_line(sys, [430, 400; 430, 320; 450, 320])
add_line(sys, [340, 330; 355, 330; 365, 390])
add_line(sys, [245, 410; 255, 490])
add_line(sys, [360, 80; 360, 145; 390, 145])
add_line(sys, [55, 80; 90, 80])
add_line(sys, [90, 20; 90, 65; 105, 65])
drawnow

% Return any arguments.
if (nargin < nargout)
    % Must use feval here to access system in memory
    if (nargin > 3)
        if (flag == 0)
            eval({'[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'})
        else
            eval({'ret =', sys,'(t,x,u,flag);'})
        end
    else
        [ret,x0,str,ts,xts] = feval(sys);
    end
    else
        drawnow % Flash up the model and execute load callback
    end
Appendix B: SIMULINK Implementation of a CDMA Transmitter Using QPSK

simq13.m

function [ret,x0,str,ts,xis]=simq13(t,x,u,flag);

% SIMQ13 is the M-file description of the SIMULINK system named SIMQ13.
% SIMQ13 has the following characteristics:
% 2 continuous states
% 0 discrete states
% 0 outputs
% 0 inputs
% does not have direct feedthrough
% 1 sample times

% The block-diagram can be displayed by typing: SIMQ13.

% SYS=SIMQ13(T,X,U,FLAG) returns depending on FLAG certain
% system values given time point, T, current state vector, X,
% and input vector, U.
% FLAG is used to indicate the type of output to be returned in SYS.
% Setting FLAG=1 causes SIMQ13 to return state derivatives, FLAG=2
% discrete states, FLAG=3 system outputs and FLAG=4 next sample
% time. For more information and other options see SFUNC.

% Calling SIMQ13 with a FLAG of zero:
% [SIZES]=SIMQ13([],[],[],0), returns a vector, SIZES, which
% contains the sizes of the state vector and other parameters.
% SIZES(1) number of states
% SIZES(2) number of discrete states
% SIZES(3) number of outputs
% SIZES(4) number of inputs
% SIZES(5) number of roots (currently unsupported)
% SIZES(6) direct feedthrough flag
% SIZES(7) number of sample times

% For the definition of other parameters in SIZES, see SFUNC.
% See also, TRIM, LIN MOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.

% Note: This M-file is only used for saving graphical information;
% after the model is loaded into memory an internal model
% representation is used.

% the system will take on the name of this mfile:
sys = mfilename;
new_system(sys)
SIMULINK Implementation of a CDMA Transmitter

```matlab
\( u (0 = (\text{nargin} + \text{nargout})) \)

```

```matlab
set_param(sys,'Location',[0,40,784,584])
open_system(sys)
end;
```

```matlab
set_param(sys,'algorithm', 'RK-45')
set_param(sys,'Start time', '0.0')
set_param(sys,'Stop time', '12')
set_param(sys,'Min step size', '0.01')
set_param(sys,'Max step size', '0.01')
set_param(sys,'Relative error','1e-6')
set_param(sys,'Return vars', "")
```

```matlab
add_block('built-in/Scope',[sys,'/','Even QPSK '])
set_param([sys,'/','Even QPSK '],...
    'Vgain','2.000000',...
    'Hgain','5.040000',...
    'Vmax','4.000000',...
    'Hmax','10.080000',...
    'Window',[2,50,791,245],...
    'position',[585,22,605,48])
```

```matlab
add_block('built-in/Note',[sys,'/','CDMA TRANSMITTER MODEL USING QPSK MODULATION'])
set_param([sys,'/','CDMA TRANSMITTER MODEL USING QPSK MODULATION'],...
    'position',[135,505,140,510])
```

```matlab
add_block('built-in/Look Up Table',[sys,'/','Even Data',13,' Mapping'])
set_param([sys,'/','Even Data',13,' Mapping'],...
    'Input_Values','[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ]',...
    'Output_Va1ues','[0 0 2 4 4 6 6 8 8 10 10 12 12 14 14 ]',...
    'position',[280,142,315,178])
```

```matlab
add_block('built-in/Sum',[sys,'/','Sum'])
set_param([sys,'/','Sum'],...
    'inputs','+-',...
    'position',[575,185,595,205])
```

```matlab
add_block('built-in/Scope',[sys,'/','QPSK'])
set_param([sys,'/','QPSK'],...
    'Vgain','1.500000',...
    'Hgain','2.500000',...
    'Vmax','3.000000',...
    'Hmax','5.000000',...
    'Window',[0,0.775,524],...
    'position',[690,182,710,208])
```

```matlab
add_block('built-in/To Workspace',[sys,'/','To Workspace1'])
```

110
SIMULINK Implementation of a CDMA Transmitter

set_param([sys,'/','To Workspace1'],...
  'mat-name','qout',...
  'buffer',100',...
  'position',[620,142,670,158])

add_block('built-in/Product',[sys,'/','Product2'])
set_param([sys,'/','Product2'],...
  'position',[680,335,705,355])

add_block('built-in/Scope',[sys,'/','Output'])
set_param([sys,'/','Output'],...
  'Vgain',1.000000',...
  'Hgain',10.000000',...
  'Vmax',2.000000',...
  'Hmax',20.000000',...
  'Window',[3,100,786,480],...
  'position',[740,332,760,358])

add_block('built-in/To Workspace',[sys,'/','To Workspace'])
set_param([sys,'/','To Workspace'],...
  'mat-name','qout',...
  'buffer',2000',...
  'position',[695,407,745,423])

add_block('built-in/Scope',[sys,'/','odd'])
set_param([sys,'/','odd'],...
  'Vgain',5.040000',...
  'Hgain',40.000000',...
  'Vmax',10.080000',...
  'Hmax',80.000000',...
  'Window',[7,204,790,417],...
  'position',[450,402,470,428])

add_block('built-in/Look Up Table',[sys,'/','Odd Data',13,'Mapping'])
set_param([sys,'/','Odd Data',13,'Mapping'],...
  'Input_Values',[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15],...
  'Output_Values',[1 1 3 5 5 7 7 9 9 11 11 13 13 15 15],...
  'position',[285,335,315,375])

add_block('built-in/Product',[sys,'/','Product1'])
set_param([sys,'/','Product1'],...
  'position',[515,300,540,320])

add_block('built-in/Scope',[sys,'/','Scope'])
set_param([sys,'/','Scope'],...
  'Vgain',1.000000',...
  'Hgain',1.000000',...
  'Vmax',2.000000',...
  'Hmax',2.000000',...
SIMULINK Implementation of a CDMA Transmitter

'Window',[112,160,443,474],...
'position',[600,296,620,324])

% Subsystem 'cos(Wct+integral[W(t)dt])'.

new_system([sys,'/','cos(Wct+integral[W(t)dt])]')
set_param([sys,'/','cos(Wct+integral[W(t)dt])]','Location',[80,202,560,377])
open_system([sys,'/','cos(Wct+integral[W(t)dt])]')

add_block('built-in/Sum',[sys,'/','cos(Wct+integral[W(t)dt])/Sum'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Sum'],...
'position',[375,65,395,85])

% Subsystem 'cos(Wct+integral[W(t)dt])/Wct'.

new_system([sys,'/','cos(Wct+integral[W(t)dt])/Wct'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct'],'Location',[346,157,531,277])
open_system([sys,'/','cos(Wct+integral[W(t)dt])/Wct'])

add_block('built-in/Product',[sys,'/','cos(Wct+integral[W(t)dt])/Wct/Product2'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct/Product2'],...
'position',[105,65,130,85])

add_block('built-in/Clock',[sys,'/','cos(Wct+integral[W(t)dt])/Wct/Clock'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct/Clock'],...
'position',[45,60,65,80])

add_block('built-in/Scope',[sys,'/','cos(Wct+integral[W(t)dt])/Wct/Scope5'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct/Scope5'],...
'Vgain','10.000000',...
'Hgain','1.000000',...
'Vmax','20.000000',...
'Hmax','2.000000',...
'Window',[100,100,422,480],...
'position',[170,22,190,48])

add_block('built-in/Outport',[sys,'/','cos(Wct+integral[W(t)dt])/Wct/out_1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct/out_1'],...
'position',[175,65,195,85])

add_block('built-in/Constant',[sys,'/','cos(Wct+integral[W(t)dt])/Wct/Wc'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct/Wc'],...
'Value','2*pi*14',...
'position',[50,107,70,123])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Wct'],[75,115;90,115;100,80])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Wct'],[70,70;100,70])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Wct'],[135,75;170,75])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Wct'],[135,75;135,35;165,35])

% Finished composite block 'cos(Wct+integral[W(t)dt])/Wct'.

set_param([sys,'/','cos(Wct+integral[W(t)dt])/Wct'],
'position',[310,10,340,60])

add_block('built-in/MATLAB Fcn',[sys,'/','cos(Wct+integral[W(t)dt])/Cos'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Cos'],
'MATLAB Fcn','cos',
'position',[435,50,485,80])

add_block('built-in/Integrator',[sys,'/','cos(Wct+integral[W(t)dt])/Integrator'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Integrator'],
'Initial','-999999',
'position',[270,105,290,125])

add_block('built-in/Scope',[sys,'/','cos(Wct+integral[W(t)dt])/Scope'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Scope'],
'Vgain','1.030000',
'Igain','28.000000',
'Vmax','2.060000',
'Hmax','56.000000',
'Window',[100,100,773,480],
'position',[270,62,290,88])

add_block('built-in/Sum',[sys,'/','cos(Wct+integral[W(t)dt])/W(t)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/W(t)'],
'position',[200,97,235,133])

add_block('built-in/Gain',[sys,'/','cos(Wct+integral[W(t)dt])/2/9*alpha'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/2/9*alpha'],
'Gain','2/9*3.72',
'position',[105,70,125,90])

add_block('built-in/Gain',[sys,'/','cos(Wct+integral[W(t)dt])/2/9*alpha*beta'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/2/9*alpha*beta'],
'Gain','2/9*3.72*0.2',
'position',[105,155,125,175])

% Subsystem 'cos(Wct+integral[W(t)dt])/Tr(t//14Tc)'.

new_system([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t//14Tc)'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t//14Tc)'],
'Location',[4,123,289,363])

add_block('built-in/Gain',[sys,'/','cos(Wct+integral[W(t)dt])/Tr(t//14Tc)/Gain1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t//14Tc)/Gain1'],
'position',[105,155,125,175])
SIMULINK Implementation of a CDMA Transmitter

'Gain', pi/14, ... 
'position', [100, 55, 120, 75]

add_block('built-in/Clock', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Clock'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Clock'], ...
'position', [60, 55, 80, 75])

add_block('built-in/Sum', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Tr(t)'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Tr(t)'], ...
'position', [230, 150, 250, 170])

add_block('built-in/Constant', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Constant'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Constant']), ...
'position', [185, 175, 205, 195])

add_block('built-in/Gain', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Gain'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Gain']), ...
'Gain', 2, ... 
'position', [185, 130, 205, 150])

add_block('built-in/Abs', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Abs'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Abs']), ...
'position', [140, 130, 165, 150])

add_block('built-in/MATLAB Fcn', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Sawtooth'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Sawtooth']), ...
'MATLAB Fcn', 'sawtooth', ...
'position', [65, 125, 115, 155])

add_block('built-in/Scope', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Scope'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Scope']), ...
'Vgain', '1.000000', ... 
'Hgain', '2.000000', ... 
'Vmax', '2.000000', ... 
'Hmax', '2.000000', ...
'Window', [100, 100, 422, 480], ...
'position', [295, 82, 315, 108])

add_block('built-in/Outport', [sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Out_1'])
set_param([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)/Out_1']), ...
'position', [270, 150, 290, 170])

add_line([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)]', [210, 185; 215, 185; 225, 165])
add_line([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)]', [210, 140; 215, 140; 225, 155])
add_line([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)]', [125, 65; 145, 65; 145, 100; 45, 1 00; 45; 140; 60, 140])
add_line([sys, '/', 'cos(Wct + integral[W(t)]dt)/Tr(t/14Tc)]', [85, 65; 95, 65])
SIMULINK Implementation of a CDMA Transmitter

add_line([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/14Tc)\]',[170,140;180,140])
add_line([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/14Tc)\]',[120,140;135,140])
add_line([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/14Tc)\]',[255,160;265,160])
add_line([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/14Tc)\]',[255,160;255,95;290,95])

% Finished composite block 'cos(Wc+\int W(t)dt)/Tr(t/14Tc)'.

set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/14Tc)\]',...
'position',[35,140,65,190])

% Subsystem 'cos(Wc+\int W(t)dt)/Tr(t/7Tc)'.

new_system([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)\'],...'Location',[4,123,289,363])

add_block('built-in/Gain',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Gain1\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Gain1\'],...
'Gain','\pi/7',...
'position',[100,55,120,75])

add_block('built-in/Clock',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Clock\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Clock\'],...
'position',[60,55,80,75])

add_block('built-in/Sum',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Tr(t)\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Tr(t)\'],...
'inputs','+-',...
'position',[230,150,250,170])

add_block('built-in/Constant',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Constant\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Constant\'],...
'position',[185,175,205,195])

add_block('built-in/Gain',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Gain\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Gain\'],...
'Gain','2',...
'position',[185,130,205,150])

add_block('built-in/Abs',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Abs\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Abs\'],...
'position',[140,130,165,150])

add_block('built-in/MATLAB Fcn',[sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Sawtooth\'])
set_param([sys,'f,\'cos(Wc+\int W(t)dt)/Tr(t/7Tc)/Sawtooth\'],...
'MATLAB Fcn','sawtooth',...
'position',[65,125,115,155])

115
SIMULINK Implementation of a CDMA Transmitter

```matlab
add_block('built-in/Scope',[sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/Scope'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/Scope'],
'Vgain',1.000000', ... 
'Hgain',2.000000', ... 
'Vmax',2.000000', ... 
'Hmax',4.000000', ... 
'Window',[163,221,499,601]', ...
'position',[295,82,315,108])

add_block('built-in/Outport',[sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)/out_1'], ...
'position',[270,150,290,170])

add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[210,185;215,185;225,165])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[210,140;215,140;225,155])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[125,65;145,65;145,100;45,100;45;140;60,140])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[85,65;95,65])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[170,140;180,140])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[120,140;135,140])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[255,160;265,160])
add_line([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'],[255,160;255,95;290,95])

set_param([sys,'/','cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'], ...
'position',[30,55,60,105])

add_block('built-in/Outport',[sys,'/','cos(Wct+integral[W(t)dt])/out_1'])
set_param([sys,'/','cos(Wct+integral[W(t)dt])/out_1'], ...
'position',[515,55,535,75])

add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[295,115;345,115;345,80;370,80])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[400,75;420,75;430,65])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[345,35;360,35;370,70])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[240,115;265,115])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[240,115;240,75;265,75])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[130,165;175,165;175,125;195,125])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[130,80;185,80;195,105])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[65,80;100,80])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[70,165;100,165])
add_line([sys,'/','cos(Wct+integral[W(t)dt)])',[490,65;510,65])

set_param([sys,'/','cos(Wct+integral[W(t)dt)])', ...
'position',[435,60,465,110])
```

% Finished composite block 'cos(Wct+integral[W(t)dt])/Tr(t/7Tc)'.

% Finished composite block 'cos(Wct+integral[W(t)dt])'.

...
SIMULINK Implementation of a CDMA Transmitter

add_block('built-in/Scope', [sys, '/', 'Even Data'])
set_param([sys, '/', 'Even Data'],
   'Vgain', '10.000000',
   'Hgain', '10.000000',
   'Vmax', '20.000000',
   'Hmax', '20.000000',
   'Window', [48, 47, 370, 312],
   'position', [350, 207, 370, 233])

add_block('built-in/Scope', [sys, '/', 'Odd Data'])
set_param([sys, '/', 'Odd Data'],
   'Vgain', '10.000000',
   'Hgain', '10.000000',
   'Vmax', '20.000000',
   'Hmax', '20.000000',
   'Window', [49, 314, 371, 545],
   'position', [335, 397, 355, 423])

% Subsystem 'PN Sequence'.
new_system([sys, '/', 'PN Sequence'])
set_param([sys, '/', 'PN Sequence'], 'Location', [12, 190, 749, 577])

% Subsystem 'PN Sequence/PN Sequence'.
new_system([sys, '/', 'PN Sequence/PN Sequence'])
set_param([sys, '/', 'PN Sequence/PN Sequence'], 'Location', [316, 263, 598, 417])

add_block('built-in/Inport', [sys, '/', 'PN Sequence/PN Sequence/yO'])
set_param([sys, '/', 'PN Sequence/PN Sequence/yO'],
   'Port', '2',
   'position', [20, 80, 40, 100])

add_block('built-in/Outport', [sys, '/', 'PN Sequence/PN Sequence/table out'])
set_param([sys, '/', 'PN Sequence/PN Sequence/table out'],
   'position', [215, 55, 235, 75])

add_block('built-in/Inport', [sys, '/', 'PN Sequence/PN Sequence/xO'])
set_param([sys, '/', 'PN Sequence/PN Sequence/xO'],
   'position', [20, 25, 40, 45])

add_block('built-in/S-Function', [sys, '/', 'PN Sequence/PN Sequence/S-function'])
set_param([sys, '/', 'PN Sequence/PN Sequence/S-function'],
   'function name', 'sftable2',
   'parameters', 'xindex, yindex, table',
   'position', [140, 52, 190, 78])
add_block('built-in/Mux', [sys,'/','PN Sequence/PN Sequence/Mux'])
set_param([sys,'/','PN Sequence/PN Sequence/Mux'], ...
    'inputs', 2, ...
    'position', [90, 46, 120, 79])
add_line([sys,'/','PN Sequence/PN Sequence'], [45, 90; 65, 90; 65, 70; 85, 70])
add_line([sys,'/','PN Sequence/PN Sequence'], [195, 65; 210, 65])
add_line([sys,'/','PN Sequence/PN Sequence'], [45, 35; 65, 35; 65, 55; 85, 55])
add_line([sys,'/','PN Sequence/PN Sequence'], [125, 65; 135, 65])
set_param([sys,'/','PN Sequence/PN Sequence'], ...
    'Mask Display', 'plot(-10,-10,II0,II0,[90,50,180],[90,40,30],[90,50,10],[50,26,20],[90,50,10],[22,13,10])', ...
    'Mask Type', '2-D Table Lookup'
set_param([sys,'/','PN Sequence/PN Sequence'], ...
    'Mask Dialogue', 'Two Dimensional Table Lookup
The first input corresponds to X Index and the second input corresponds to the Y Index
IndexTable'
set_param([sys,'/','PN Sequence/PN Sequence'], ...
    'Mask Translate', 'xindex=@1; yindex=@2; table=@3;
sftab2chk(xindex,yindex,table);')
set_param([sys,'/','PN Sequence/PN Sequence'], ...
    'Mask Help', 'This block returns a linearly interpolated intersection from the table using the X index (which corresponds to the rows of the table) and the Y index (which corresponds to the columns of the table). Extrapolation is used.'
set_param([sys,'/','PN Sequence/PN Sequence'], ...
    'Mask Entries', ['1 2 3 4 5 6 7 8 9']
set_param([sys,'/','PN Sequence/PN Sequence'], ...
    'Vgain', '5.000000', ...
    'Hgain', '5.000000', ...
    'Vmax', '10.000000', ...
    'Hmax', '10.000000', ...
    'Window', [39, 356, 361, 568], ...
    'position', [430, 152, 450, 178])
add_block('built-in/Outport', [sys,'/','PN Sequence/out_1'])
set_param([sys,'/','PN Sequence/out_1'], ...
    'position', [555, 45, 575, 65])
add_block('built-in/Scope', [sys,'/','PN Sequence/PN1'])
set_param([sys,'/','PN Sequence/PN1'], ...)
SIMULINK Implementation of a CDMA Transmitter

add_line([sys,'/','PN Sequence'],[420,60;455,60;465,85])
add_line([sys,'/','PN Sequence'],[520,95;520,165;540,165])
add_line([sys,'/','PN Sequence'],[415,105;425,165])
add_line([sys,'/','PN Sequence'],[415,105;465,105])
add_line([sys,'/','PN Sequence'],[520,95;540,95;550,55])
add_line([sys,'/','PN Sequence'],[170,105;200,105])
add_line([sys,'/','PN Sequence'],[95,165;115,165;115,110;130,110])
add_line([sys,'/','PN Sequence'],[95,100;130,100])
add_line([sys,'/','PN Sequence'],[250,105;355,105])
add_line([sys,'/','PN Sequence'],[310,105;310,155;250,155])

% Finished composite block 'PN Sequence'.
set_param([sys,'/','PN Sequence'],
'position',[425,450,455,500])

% Subsystem ['Odd Data ',13,'Sequence'].
new_system([sys,'/',['Odd Data ',13,'Sequence']])
set_param([sys,'/',['Odd Data ',13,'Sequence']],'Location',[316,263,598,417])
add_block('built-in/Inport',[sys,'/',['Odd Data ',13,'Sequence/y0']])
set_param([sys,'/',['Odd Data ',13,'Sequence/y0']],
'Port',2,
'position',[20,80,40,100])
add_block('built-in/Outport',[sys,'/',['Odd Data ',13,'Sequence/table out']])
set_param([sys,'/',['Odd Data ',13,'Sequence/table out']],
'position',[215,55,235,75])
add_block('built-in/Inport',[sys,'/',['Odd Data ',13,'Sequence/x0']])
set_param([sys,'/',['Odd Data ',13,'Sequence/x0']],
'position',[20,25,40,45])
add_block('built-in/S-Function',[sys,'/',['Odd Data ',13,'Sequence/S-function']])
set_param([sys,'/',['Odd Data ',13,'Sequence/S-function']],
'function name','sftable2',
'parameters','xindex, yindex, table',
'position',[140,52,190,78])
add_block('built-in/Mux',[sys,'/',['Odd Data ',13,'Sequence/Mux']])
set_param([sys,'/',['Odd Data ',13,'Sequence/Mux']],
'inputs',2,
'position',[90,46,120,79])
add_line([sys,'/',['Odd Data ',13,'Sequence']],[45,90;65,90;65,70;85,70])
add_line([sys,'/',['Odd Data ',13,'Sequence']],[195,65;210,65])
add_line([sys,'/',['Odd Data ',13,'Sequence']],[45,35;65,35;65,55;85,55])
SIMULINK Implementation of a CDMA Transmitter

add_line([sys,'/','Odd Data',13,'Sequence'],[125,65;135,65])
set_param([sys,'/','Odd Data',13,'Sequence'],...
'Mask Display','plot(-10,-10,110,[90,50,10],[90,40,30],[90,50,10],[50,26,20],[90,50,10],[22,13,10])',...
'Mask Type','2-D Table Lookup')
set_param([sys,'/','Odd Data',13,'Sequence'],...
'Mask Dialogue','Two Dimensional Table Lookup')
The first input corresponds to X Index and the second input corresponds to the Y Index
set_param([sys,'/','Odd Data',13,'Sequence'],...
'Mask Translate','xindex=@1; yindex=@2; table=@3;
sftab2chk(xindex,yindex,table)')
set_param([sys,'/','Odd Data',13,'Sequence'],...
'Mask Help','This block returns a linearly interpolated intersection from
the table using the X index (which corresponds to the rows of the table) and the Y
index (which corresponds to the columns of the table). Extrapolation is used.')
set_param([sys,'/','Odd Data',13,'Sequence'],...
'Mask Entries','[1 2 3 4 5 6 7 8 9]V[I 3 5 7 9 11 13 15]V[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]V')

% Finished composite block 'Odd Data',13,'Sequence'.
set_param([sys,'/','Odd Data',13,'Sequence'],...
'position',[380,338,410,362])

% Subsystem 'sin(Wct+integral[W(t)dt])'.
new_system([sys,'/','sin(Wct+integral[W(t)dt])'])
set_param([sys,'/','sin(Wct+integral[W(t)dt])'],'Location',[80,202,617,415])
add_block('built-in/Sum',[sys,'/','sin(Wct+integral[W(t)dt])/Sum'])
set_param([sys,'/','sin(Wct+integral[W(t)dt])/Sum'],...
'position',[375,65,395,85])

% Subsystem 'sin(Wct+integral[W(t)dt])/Wct'.
new_system([sys,'/','sin(Wct+integral[W(t)dt])/Wct'])
set_param([sys,'/','sin(Wct+integral[W(t)dt])/Wct'],'Location',[345,138,584,346])
add_block('built-in/Product',[sys,'/','sin(Wct+integral[W(t)dt])/Wct/Product2'])
set_param([sys,'/','sin(Wct+integral[W(t)dt])/Wct/Product2'],...
'position',[105,65,130,85])
add_block('built-in/Clock',[sys,'/','sin(Wct+integral[W(t)dt])/Wct/Clock'])
set_param([sys,'/','sin(Wct+integral[W(t)dt])/Wct/Clock'],...
SIMULINK Implementation of a CDMA Transmitter

'position',[45,60,65,80])

add_block('built-in/Scope', [sys,'/','sin(Wct+integral[W(t)dt]) /Wct/Scope5'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Wct/Scope5'],
'Vgain','10,000000',
'Hgain','1.000000',
'Vmax','20.000000',
'Hmax','2.000000',
'Window',[100,100,422,480],
'position',[170,22,190,48])

add_block('built-in/Outport', [sys,'/','sin(Wct+integral[W(t)dt]) /Wct/out_1'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Wct/out_1'],
'position',[175,65,195,85])

add_block('built-in/Constant', [sys,'/','sin(Wct+integral[W(t)dt]) /Wct/We'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Wct/We'],
'Value','2*pi*14',
'position',[50,100,75,130])

add_line([sys,'/','sin(Wct+integral[W(t)dt]) /Wct'],[80,115;90,115;100,80])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) /Wct'],[70,70;100,70])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) /Wct'],[135,75;170,75])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) /Wct'],[135,75;165,35])

% Finished composite block 'sin(Wct+integral[W(t)dt]) /Wct'.

set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Wct'],
'position',[310,10,340,60])

add_block('built-in/MATLAB Fcn', [sys,'/','sin(Wct+integral[W(t)dt]) /Sin'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Sin'],
'position',[435,50,485,80])

add_block('built-in/Integrator', [sys,'/','sin(Wct+integral[W(t)dt]) /Integrator'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Integrator'],
'Initial','-9999999',
'position',[270,105,290,125])

add_block('built-in/Scope', [sys,'/','sin(Wct+integral[W(t)dt]) /Scope'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Scope'],
'Vgain','1.030000',
'Hgain','28.000000',
'Vmax','2.060000',
'Hmax','56.000000',
'Window',[100,100,773,480],
'position',[270,62,290,88])

add_block('built-in/Sum', [sys,'/','sin(Wct+integral[W(t)dt]) /W(t)'])
SIMULINK Implementation of a CDMA Transmitter

```matlab
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /W(t)'],
    'position',[200,97,235,133])

add_block('built-in/Gain',[sys,'/','sin(Wct+integral[W(t)dt]) /2/9*alpha'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /2/9*alpha'],
    'Gain','2/9*3.72',
    'position',[105,70,125,90])

add_block('built-in/Gain',[sys,'/','sin(Wct+integral[W(t)dt]) /2/9*alpha*beta'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /2/9*alpha*beta'],
    'Gain','2/9*3.72*0.2',
    'position',[105,155,125,175])

% Subsystem 'sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)'.
new_system([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)'],'Location',[4,123,289,363])

add_block('built-in/Gain',[sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Gain1'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Gain1'],
    'Gain','pi/14',
    'position',[100,55,120,75])

add_block('built-in/Clock',[sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Clock'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Clock'],
    'position',[60,55,80,75])

add_block('built-in/Sum',[sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Tr(t)'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Tr(t)'],
    'inputs','+',
    'position',[230,150,250,170])

add_block('built-in/Constant',[sys,'/','sin(Wct+integral[W(t)dt])
/Tr(t/14Tc)/Constant'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Constant'],
    'position',[185,175,205,195])

add_block('built-in/Gain',[sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Gain1'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Gain1'],
    'Gain','2',
    'position',[185,130,205,150])

add_block('built-in/Abs',[sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Abs'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /Tr(t/14Tc)/Abs'],
    'position',[140,130,165,150])

add_block('built-in/MATLAB Fcn',[sys,'/','sin(Wct+integral[W(t)dt])
/Tr(t/14Tc)/Sawtooth'])
```

123
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)/Sawtooth']...,
'MATLAB Fcn', 'sawtooth', ...,
'position', [65,125,115,155])

add_block('built-in/Scope', [sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)/Scope'])
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)/Scope'], ...
'Vgain', '1.000000', ...,
'Hgain', '2.000000', ...,
'Vmax', '2.000000', ...,
'Hmax', '2.000000', ...,
'Window', [100,100,422,480], ...,
'position', [295,82,315,108])

add_block('built-in/Outport', [sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)/out_1'])
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)/out_1'], ...
'position', [270,150,290,170])

add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [210,185;215,185;225,165])
add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [210,140;215,140;225,155])
add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [125,65;145,65;145,100;45,100;45,140;60,140])
add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [85,65;95,65])
add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [120,140;135,140])
add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [255,160;265,160])
add_line([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], [255,160;255,95;290,95])

% Finished composite block 'sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'.

set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/14Tc)'], ...
'position', [35,140,65,190])

% Subsystem 'sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)'.

new_system([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)'])
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)'], 'Location', [4,123,289,363])

add_block('built-in/Gain', [sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)/Gain1'])
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)/Gain1'], ...
'Gain', 'pi/7', ...
'position', [100,55,120,75])

add_block('built-in/Clock', [sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)/Clock'])
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)/Clock'], ...
'position', [60,55,80,75])

add_block('built-in/Sum', [sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)/Tr(t)'])
set_param([sys,'/','sin(Wct+integral(W(t)dt)) /Tr(t/7Tc)/Tr(t)'], ...)
SIMULINK Implementation of a CDMA Transmitter

'inputs', '+', '…
'position', [230, 150, 250, 170])

add_block('built-in/Constant', [sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Constant])
set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Constant], ...
'position', [185, 175, 205, 195])

add_block('built-in/Gain', [sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Gain])
set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Gain], ...
'Gain', '2', ...
'position', [185, 130, 205, 150])

add_block('built-in/Abs', [sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Abs])
set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Abs], ...
'position', [140, 130, 165, 150])

add_block('built-in/MATLAB Fcn', [sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Sawtooth])
set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Sawtooth], ...
'MATLAB Fcn', 'sawtooth', ...
'position', [65, 125, 115, 155])

add_block('built-in/Scope', [sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Scope])
set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/Scope], ...
'Vgain', '1.000000', ...
'Hgain', '2.000000', ...
'Vmax', '2.000000', ...
'Hmax', '4.000000', ...
'Window', [163, 221, 499, 601], ...
'position', [295, 82, 315, 108])

add_block('built-in/Outport', [sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1])
set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], ...
'position', [270, 150, 290, 170])

add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [210, 185, 215, 185, 225, 165])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [210, 140, 215, 140, 225, 155])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [125, 65, 145, 65, 145, 100, 45, 100, 45, 140, 60, 140])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [85, 65, 95, 65])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [170, 140, 180, 140])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [120, 140, 135, 140])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [255, 160, 265, 160])
add_line([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)/out_1], [255, 160, 255, 95, 290, 95])

% Finished composite block 'sin(Wct+integral[W(t)dt])/Tr(t/Tc)'.

set_param([sys, '/', sin(Wct+integral[W(t)dt])/Tr(t/Tc)], ...
'position', [30, 55, 60, 105])

125
SIMULINK Implementation of a CDMA Transmitter

% Function: {sin(Wct+integral[W(t)dt]) /out_1}

add_block('built-in/Outport', [sys,'/','sin(Wct+integral[W(t)dt]) /out_1'])
set_param([sys,'/','sin(Wct+integral[W(t)dt]) /out_1'],
'position',[515,55,535,75])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [295,115;345,115;345,80;370,80])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [400,75;420,75;430,65])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [345,35;360,35;370,70])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [240,115;265,115])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [240,115;240,75;265,75])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [130,165;175,165;175,125;195,125])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [130,80;185,80;195,105])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [65,80;100,80])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [70,165;100,165])
add_line([sys,'/','sin(Wct+integral[W(t)dt]) '], [490,65;510,65])

% Finished composite block 'sin(Wct+integral[W(t)dt])'.

set_param([sys,'/','sin(Wct+integral[W(t)dt]) '],
'position',[410,245,440,295])

% Subsystem 'Even Data Sequence'.

new_system([sys,'/','Even Data Sequence'])
set_param([sys,'/','Even Data Sequence'],'Location',[316,263,598,417])

add_block('built-in/Inport', [sys,'/','Even Data Sequence/y0'])
set_param([sys,'/','Even Data Sequence/y0'],
'Port','2',
'position',[20,80,40,100])

add_block('built-in/Outport', [sys,'/','Even Data Sequence/table out'])
set_param([sys,'/','Even Data Sequence/table out'],
'position',[215,55,235,75])

add_block('built-in/Inport', [sys,'/','Even Data Sequence/x0'])
set_param([sys,'/','Even Data Sequence/x0'],
'position',[20,25,40,45])

add_block('built-in/S-Function', [sys,'/','Even Data Sequence/S-function'])
set_param([sys,'/','Even Data Sequence/S-function'],
'function name','sftable2',
'parameters','xindex, yindex, table',
'position',[140,52,190,78])

add_block('built-in/Mux', [sys,'/','Even Data Sequence/Mux'])
set_param([sys,'/','Even Data Sequence/Mux'],
'inputs','2',
'position',[140,52,190,78])
SIMULINK Implementation of a CDMA Transmitter

'SIMULINK' implementation of a CDMA transmitter.

add_line([sys,'/','Even Data Sequence'],[45,90;65,90;65,70;85,70])
add_line([sys,'/','Even Data Sequence'],[195,65;210,65])
add_line([sys,'/','Even Data Sequence'],[45,35;65,35;65,55;85,55])
add_line([sys,'/','Even Data Sequence'],[125,65;135,65])

set_param([sys,'/','Even Data Sequence'],...
'Mask Display','plot(-10,10,110,110,[90,50,10],[90,40,30],[90,50,10],[50,26,20],[90,50,10],[22,13,10]',...
'Mask Type','2-D Table Lookup')

set_param([sys,'/','Even Data Sequence'],...
'Mask Dialogue','Two Dimensional Table Lookup: The first input corresponds to the X index and the second input corresponds to the Y index. (Index X Index Y Index Table)'

set_param([sys,'/','Even Data Sequence'],...
'Mask Translate','xindex=@1; yindex=@2; table=@3; sftab2chk(xindex, yindex, table);')

set_param([sys,'/','Even Data Sequence'],...
'Mask Help','This block returns a linearly interpolated intersection from the table using the X index (which corresponds to the rows of the table) and the Y index (which corresponds to the columns of the table). Extrapolation is used. '

set_param([sys,'/','Even Data Sequence'],...
'Mask Entries','[1 2 3 4 5 6 7 8 9]V[0 2 4 6 8 10 12 14]V[1 1 1 1 1 1 1 1 1; 1 1 -1 1 1 -1 1 1 1; 1 1 1 -1 1 1 1 1 1; 1 1 1 1 1 1 1 1 1; -1 1 1 1 1 1 1 1 1; -1 1 1 1 1 1 1 1 1; 1 -1 1 1 1 1 1 1 1; 1 -1 1 1 1 1 1 1 1]V')

set_param([sys,'/','Even Data Sequence'],...
'position',[360,143,390,167])

add_block('built-in/Product',[sys,'/','Product'])
set_param([sys,'/','Product'],...
'position',[505,140,530,160])

add_block('built-in/Scope',[sys,'/','even'])
set_param([sys,'/','even'],...
'Vgain','5.040000',...
'Hgain','40.000000',...
'Vmax','10.080000',...
'Hmax','80.000000',...
'Window','[3,44,787,235]',...
'position','[410,12,430,38]')

add_block('built-in/Constant',[sys,'/','Even Data Row'])
set_param([sys,'/','Even Data Row'],...
'Value','3',...
'position',[310,90,330,110])
add_block('built-in/Constant', [sys, '/', 'Odd Data Row'])
set_param([sys, '/', 'Odd Data Row'], ...
    'Value', '3', ...
    'position', [305.289, 325, 311])

add_block('built-in/Clock', [sys, '/', 'Clock'])
set_param([sys, '/', 'Clock'], ...
    'position', [120.48, 145, 72])

add_block('built-in/Constant', [sys, '/', 'Data Rate', 13, ']')
set_param([sys, '/', 'Data Rate', 13, ']'), ...
    'position', [10, 95, 30, 115])

add_block('built-in/Clock', [sys, '/', 'Clock1'])
set_param([sys, '/', 'Clock1'], ...
    'position', [25, 170, 40, 190])

add_block('built-in/Product', [sys, '/', 'Product3'])
set_param([sys, '/', 'Product3'], ...
    'position', [70, 148, 100, 172])

add_block('built-in/MATLAB Fcn', [sys, '/', 'MATLAB Fcn'])
set_param([sys, '/', 'MATLAB Fcn'], ...
    'MATLAB Fcn', 'fix', ...
    'position', [180, 145, 230, 175])

add_block('built-in/Fcn', [sys, '/', 'Fcn'])
set_param([sys, '/', 'Fcn'], ...
    'Expr', 'rem(u, 16)', ...
    'position', [120, 150, 160, 170])

add_block('built-in/Scope', [sys, '/', 'Data'])
set_param([sys, '/', 'Data'], ...
    'Vgain', '10.000000', ...
    'Hgain', '2.000000', ...
    'Vmax', '20.000000', ...
    'Hmax', '4.000000', ...
    'Window', [39, 373, 361, 568], ...
    'position', [250, 222, 270, 248])

add_block('built-in/Mux', [sys, '/', 'Mux'])
set_param([sys, '/', 'Mux'], ...
    'inputs', '3', ...
    'position', [130, 339, 160, 371])

add_line(sys, [235, 160; 275, 160])
add_line(sys, [320, 160; 355, 160])
add_line(sys, [235, 160; 245, 235])
add_line(sys, [395, 155; 500, 155])
add_line(sys, [415, 350; 500, 350; 510, 315])
add_line(sys,[425,350,425,415;445,415])
add_line(sys,[535,150;550,150;550,190;570,190])
add_line(sys,[600,195;685,195])
add_line(sys,[550,150;550,35;580,35])
add_line(sys,[265,160;265,355;280,355])
add_line(sys,[320,355;375,355])
add_line(sys,[320,355;330,410])
add_line(sys,[335,160;345,220])
add_line(sys,[710,345;735,345])
add_line(sys,[650,195;650,340;675,340])
add_line(sys,[725,345;725,385;680,385;690,415])
add_line(sys,[600,195;600,150;615,150])
add_line(sys,[460,475;530,475;530,350;675,350])
add_line(sys,[545,310;555,310;555,200;570,200])
add_line(sys,[395,155;405,25])
add_line(sys,[545,310;595,310])
add_line(sys,[445,270;500,270;510,305])
add_line(sys,[470,85;480,85;480,145;500,145])
add_line(sys,[335,100;345,100;355,150])
add_line(sys,[330,300;360,300;360,345;375,345])
add_line(sys,[165,160;175,160])
add_line(sys,[35,105;55,105;65,155])
add_line(sys,[45,180;55,180;65,165])
add_line(sys,[105,160;115,160])
drawnow

% Return any arguments.
if (nargin > 1 | nargout)
    % Must use feval here to access system in memory
    if (nargin > 3)
        if (flag == 0)
            eval(['[ret,x0,str,ts,xts] = ',sys,'(t,x,u,flag);'])
        else
            eval(['ret = ',sys,'(t,x,u,flag);'])
        end
    else
        [ret,x0,str,ts,xts] = feval(sys);
    end
else
    drawnow % Flash up the model and execute load callback
end