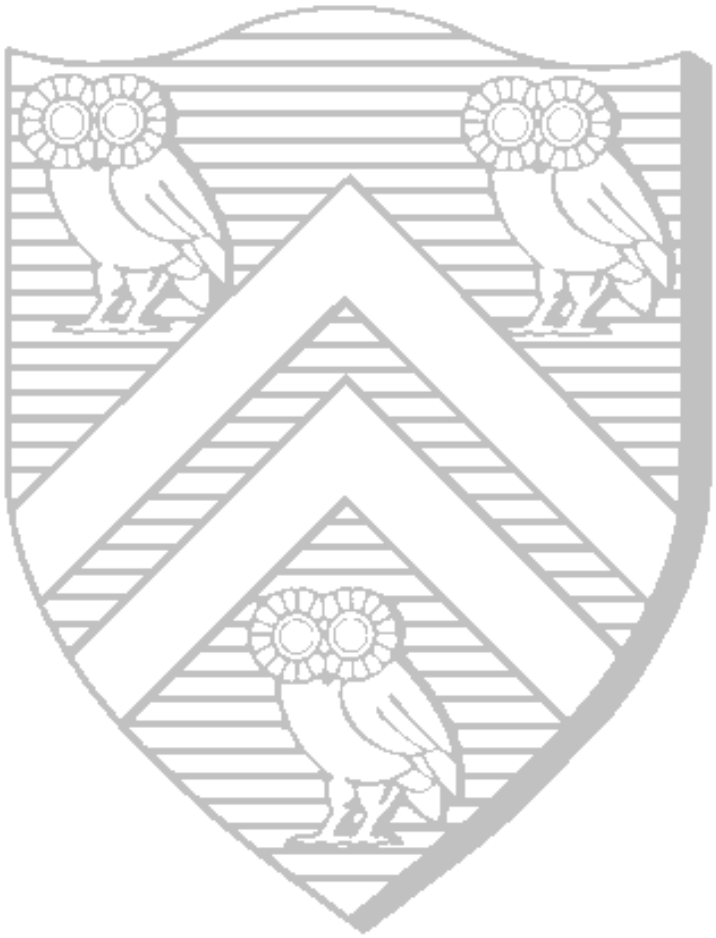
A Software Simulation Testbed for CDMA Wireless Communication Systems

Vishwas Sundaramurthy



Thesis: Master of Science
Electrical and Computer Engineering
Rice University, Houston, Texas (May 1999)

RICE UNIVERSITY

A Software Simulation Testbed for CDMA Wireless Communication Systems

by

Vishwas Sundaramurthy

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Master of Science

Approved, Thesis Committee:

Dr. Joseph R. Cavallaro, Chair
Associate Professor of Electrical and
Computer Engineering

Dr. Behnaam Aazhang
Professor of Electrical and Computer
Engineering

Dr. Peter Varman
Associate Professor of Electrical and
Computer Engineering

Houston, Texas

May, 1999

ABSTRACT

A Software Simulation Testbed for CDMA Wireless Communication Systems

by

Vishwas Sundaramurthy

This thesis develops a software wireless communication testbed which simulates a Code Division Multiple Access(CDMA) link. CDMA is a popular technology in cellular systems due to its superior capacity and performance. The designer of CDMA systems has a wide array of signal processing algorithms to choose from and a variety of operating environments to deal with. The testbed is a tool to evaluate these design options and trade-offs in different scenarios.

The backbone of this system is a wireless CDMA multiuser link built using Simulink and Matlab. An efficient method of system modeling is used to speed up the simulations. This testbed can be used to study the performance of a CDMA wireless link with variation in system parameters and channel conditions.

The use of Real-Time Workshop for DSP code generation is studied. This method is useful for rapid prototyping of algorithms and in DSP-based simulation acceleration of the testbed.

Acknowledgments

I would like to thank my advisor, Dr. Joseph Cavallaro, for all the guidance, encouragement and support. I would also like to thank Dr. Behnaam Aazhang for his encouragement and suggestions. Thanks to Dr. Peter Varman for serving on the thesis committee and for his comments. I also wish to express my gratitude to Dr. Jan Hewitt for helping me learn a thing or two about elegant writing.

The members of our research group have been very cooperative in helping with my work. My special thanks to Suman, Andrew, Chaitali, Srikrishna, Gang Xu, Dinesh, and Sridhar.

My stay at Rice has been very enjoyable due to our cooking group (for reasons other than just as a source of nourishment!). These friends have been great company and have had an influence on a lot of things in my life, from my choice of music to my taste for food (and my concept of vegetarianism!).

Finally, I wish to thank my parents, Vasantha and R. Sundaramoorthy, who have moulded me into what I am, and who will be thinking about me every passing moment.

The work documented in this thesis was supported by Nokia Corporation and by Texas Instruments. Mathworks Inc. provided an advanced version of their Matlab software system with Simulink and Real-Time Workshop, which has been extensively used in this research.

Contents

Abstract ii

Acknowledgments iii

List of Illustrations vii

List of Tables ix

1 Introduction 1

1.1 The need for a simulation testbed 2

1.2 Contributions of this thesis 3

1.3 Overview 5

2 Background 7

2.1 Wireless cellular systems and the CDMA technique 7

2.1.1 Components of a wireless cellular link 7

2.1.2 Uplink and downlink channels and system configurations . . . 13

2.2 System simulation of CDMA wireless networks 14

2.2.1 Requirements for simulating a CDMA system 14

2.2.2 State-of-the-art work in CDMA wireless communications and the need for an alternative method of simulation 15

2.3 Block diagram type simulation and modeling environments 16

3 Simulation System for CDMA Wireless Links 18

3.1 Construction of the Simulink based system 18

3.1.1 Features of the multiuser physical layer link 19

3.1.2 System development using Simulink 23

3.1.3 Global management of simulation time-steps in Simulink . . . 24
3.2 Features of the simulation system . . . 24
3.2.1 Modular simulation system with a DS-CDMA library . . . 24
3.2.2 Flexible system . . . 27
3.2.3 Automated net-list generation . . . 28
3.3 Capabilities of the system . . . 28

4 An Efficient Method of Simulation 31
4.1 Analysis of the chip-matched filtering . . . 32
4.2 An efficient method of simulation modeling . . . 35
4.3 Using the efficient method with quantized delays to capture the effect of sampling . . . 39
4.4 Comparison of simulation time for the two methods of modeling . . . 40
4.4.1 Execution time comparison in the stand-alone Matlab simulation 42
4.4.2 Simulation time comparison for the entire Simulink system . . 43
4.4.3 Profiling analysis for the Simulink based system . . . 44

5 Rapid Prototyping and Simulation Acceleration Using DSPs 47
5.1 Prototyping with Real-Time Workshop . . . 47
5.2 DSP rapid prototyping . . . 48
5.3 Simulation acceleration using stand-alone simulators . . . 48
5.4 RTW based simulation system for the Pentium host . . . 49

6 Future Work 51
6.1 Enhancing the library of blocks . . . 51
6.2 DSP based simulation acceleration . . . 52
6.3 Multi-Processing . . . 52

A **54**

A.1 Listing of system parameters and signals 54

A.2 Analysis for channel output and the subsequent joint synchronization and detection 54

A.2.1 Analysis of the data generation, effect of the channel and the chip matched filtering 55

A.2.2 Channel parameter estimation and detection 60

B **63**

B.1 Construction details of the Simulink based simulation system 63

B.1.1 Automated netlist generation 63

B.1.2 Data generation, spreading and channel modeling 64

B.1.3 Maximum Likelihood synchronization 65

B.1.4 Detector blocks 67

B.2 Profiling 69

C **70**

C.1 Board support for C-Code generation with Real Time Workshop . . . 70

Bibliography **72**

Illustrations

2.1 A generic wireless link 8
2.2 The physical layer in the CDMA scheme 9
2.3 Chipping or spreading sequence 10
2.4 Uplink and downlink 13
2.5 Dataflow driven simulation vs. cycle-based simulation 17

3.1 The basic multiuser CDMA link in Simulink 19
3.2 The utilization of different libraries 23
3.3 The simulink version of the software CDMA testbed 25
3.4 The simulink CDMA library 26
3.5 The graphical user interface 27
3.6 Automated netlist generation 28
3.7 An example run showing the bit error rate comparison of 2 users . . . 29
3.8 A phone call in a CDMA wireless network 30

4.1 The two methods of modeling – The big picture 32
4.2 The part of the system under consideration 33
4.3 The traditional method of modeling in Simulink (with sampled signals) 34
4.4 The integration methods in the chip-matched filtering 36
4.5 Comparison of the two methods of modeling 37
4.6 Received signal – *chip* level details 37
4.7 Superimposition of chip streams of 3 users (with 7 chips per bit) . . . 38

4.8 The efficient method of modeling in Simulink 39
4.9 Sampling mechanism in chip-matched filtering 41
4.10 Execution time plot for different bit-stream lengths: L $\in$ [50,1200] . 42
4.11 Execution time with different number of users : K $\in$ [2,30] 43
4.12 Comparison of simulation time with different bit-stream lengths (5 users) 44

5.1 A base-station receiver 48
5.2 The RTW based simulation (stand-alone simulator on Pentium) . . . 50

B.1 Construction of the data generation block with general spreading and arbitrary delays 64
B.2 The multipath AWGN channel 65
B.3 The delay block for the multipath channel 66
B.4 The chip matched filter 66
B.5 The channel estimation block 67
B.6 The multiuser detector block 68
B.7 The triggered part of the multiuser detector 68

C.1 The simulink block diagram for matrix multiplication 70
C.2 The underlying C-MEX S-function for matrix multiplication 71

Tables

4.1 Comparison of profiling data for the system with a decorrelating detector. 45
4.2 Comparison of profiling data for the system with a multistage detector. 46

A.1 Notations for system parameters. 54
A.2 Notations for system signals. 55

Chapter 1

Introduction

Wireless cellular telephony has been growing at a faster rate than wired-line telephone networks [1]. The main factor driving this tremendous growth in wireless coverage is that it does not need the setting up of expensive infrastructure like copper or fiber lines and switching equipment. This growth has also been fueled by the recent improvements in the capacity of wireless links due to the use of *multiple access techniques* (which allow many users to share the same channel for transmission) in association with advanced signal processing algorithms. Code Division Multiple Access (CDMA) is becoming a popular technology for cellular communications [2]. Unlike other multiple access techniques such as Frequency Division Multiple Access (FDMA) and Time-Division Multiple Access (TDMA), which are limited in frequency band and time duration respectively, CDMA uses all of the available *time-frequency space*. One form of CDMA called Direct Sequence CDMA (DS-CDMA) uses a set of unique *signature sequence or spreading codes* to modulate the data bits of different users. With the knowledge of these spreading codes, the receiver can isolate the data corresponding to each user by the process of channel estimation and detection. This process spreads the bandwidth of the underlying data signal, hence CDMA is called a *spread spectrum* technique. Standards such as IS-95 and the proposed W-CDMA are based on CDMA technology.

1.1 The need for a simulation testbed

The push for better usage of available bandwidth has driven the research in developing new signal processing algorithms for different components of the system. The developments in the physical layer receiver structure have been both in the synchronizer or detector (which extract the bits from the baseband signal from the channel) and in the coding schemes (which make efficient use of available bandwidth and protect against errors).

The designer of such a system has a wide array of algorithms and configurations to choose from. A simulation testbed for wireless communications allows the evaluation of the different choices in such a design [3–5]. There are other variables in the system, such as the properties of the channel, which are affected by environmental factors. A complete algorithmic evaluation needs to consider all such factors. The envisaged wireless testbed has the following capabilities:

- **A comprehensive method of algorithm evaluation**

  A new algorithm is typically put to use by first doing a localized evaluation (using accepted values for some of the system parameters and channel properties) and then prototyping it. This method does not provide accurate insights into how the algorithm behaves in a real system until it is actually used in one. The CDMA wireless testbed hopes to fill in this gap by providing an environment which models all facets of a real wireless system and provides a complete picture of system behavior with various algorithms.

- **Generation of performance indicators**

  Long bit-streams are passed through the simulated communication system and the errors in the received bit-stream are counted. The bits are grouped into

frames, the exact specification of which depends on the standards being used. The performance of the system is quantified using the average *bit error rate (BER)* and *frame error rate (FER)*.

- **Study of algorithm trade-off issues**

  One of the main uses of this testbed is to explore the effects of using different combinations of algorithms in the simulations. Different algorithms for channel parameter estimation, detection and coding blocks in the receiver can be studied in various configurations. Each algorithm differs in the computational complexity and in the resulting bit error rate. The testbed is intended to assist in the choice of an algorithm, that results in a trade-off between performance and implementation complexity.

- **Utilization of DSP hardware in communication systems**

  Digital Signal Processors (DSPs) are well suited for signal processing in communication applications and are extensively used in developing products like Cellular phones and Base-stations. The testbed can be used as a platform to test the use of DSPs in CDMA systems. Development boards with DSPs are typically used for these studies. There needs to be support for downloading the relevant parts of the execution to the DSP boards. This support for using DSPs in the testbed will also facilitate simulation acceleration.

1.2 Contributions of this thesis

This thesis presents the development of a software version of a Wireless CDMA testbed. A complete multiuser CDMA link is developed using Simulink and Matlab environment. This version of the testbed is used to develop efficient modeling meth-

ods for simulating CDMA systems. The testbed also showcases the work of several members in the research team at the Center for Multimedia Communications (CMC) at Rice university [6]. The thesis develops and explains the following capabilities of the simulation testbed:

- **A physical layer multiuser CDMA wireless link: the backbone of the simulation system**

  The multiuser link developed in the software simulation system is intended as a backbone for developing more complex systems. This flexible link has a user interface which allows a system designer to test different scenarios by changing the parameters (such as number of users, paths, channel properties and spreading waveforms). A library of algorithms with blocks for all parts of the system is developed. The different configurations of a CDMA wireless link can be studied using the components of the library. The modular design of the simulation system allows easy addition of new algorithms to the library.

- **An efficient method of simulation**

  The thesis introduces a method of simulation which provides accurate simulation even while sampling the channel at a lower rate than the traditional method of modeling such a system. In a real system the continuous time signal at the output of the channel is converted to a discrete time signal by sampling the output of a filter matched to the chip waveform. However, the simulation system models the channel output as a discrete time signal; hence the output of the chip matched filter has to be derived from this discrete time channel output (which is available as a finite number of samples per chip). The traditional CDMA *simulation* system has the minimum sampling time equal to a fraction

of the chip duration. The use of a discrete channel output also introduces an error in the chip matched filter output, which can be abated by increasing the channel sampling rate. But an increase in the sampling rate slows down the simulation. We incorporate a novel method of combined spreading, channel modeling and matched filtering to obtain accurate results, while keeping the system sampling time to once a chip.

- **DSP hardware prototyping support from Simulink**

  The work in this thesis develops some prototyping support from Simulink to generate the executable (simulator) for the DSPs. This compiled simulator method, even if it is running on the host processor, is faster than using the Matlab/Simulink environment, which is an interpreted system.

1.3 Overview

The rest of the thesis is divided into 5 chapters. Chapter 2 introduces in more detail the CDMA technique and the components of a typical CDMA wireless system. This chapter also introduces the simulation techniques and tools for such systems, which involve a substantial amount of signal and data processing. The features of simulation and modeling environments like Simulink and Signal Processing Worksystem (SPW) are discussed and contrasted.

The construction of the simulation framework using Simulink is described in chapter 3. The main features of the flexible multiuser physical layer link are presented. This chapter also describes the features of the algorithms for channel estimation and detection.

Chapter 4 develops a new method of modeling which increases the simulation

speed by an order of magnitude. The traditional method of simulating a CDMA system requires samples from the channel at the sub-chip level to effectively model the system behavior. A unique method of data generation is used in the system, along with some associated signal processing in the receiver to obviate the need for using data samples at a very fine granularity.

Chapter 5 describes a method in which algorithmic ideas can be quickly translated into products on DSP hardware. This process, called rapid prototyping, uses a combination of software tools and hardware. This chapter also discusses the utilization of DSPs for advanced signal processing algorithms in wireless communications.

Chapter 6 discusses the extensions possible to this system. The CDMA library can be enhanced to include a bigger set of algorithms for channel estimation, detection and coding. We also discuss the possible use of DSPs in a multi-processor configuration for simulation acceleration.

Chapter 2

Background

2.1 Wireless cellular systems and the CDMA technique

A wireless link using a multiple access scheme in a cellular system is shown in Figure 2.1. This generic link has many users modulating their bit sequences and sending them across the channel. The analog signal from the channel is processed in the receiver and the bits are detected. The following subsection discusses the different components of a typical wireless system and also introduces the Code Division Multiple Access (CDMA) Scheme.

2.1.1 Components of a wireless cellular link

The generic wireless link consisting of the transmitter, the channel and the receiver in the physical layer is described here. Typical blocks on the **transmitter** side are

1. **Source Coding:** It has been observed that data such as voice and video have redundant information in them. The bandwidth required for such data can be reduced by source coding (or compression).

2. **Channel Coding:** The data transmitted across the channel is susceptible to errors due to interference from other users, thermal noise, fading (time-varying amplitude response of the channel) and other effects. It is possible to detect and correct some of the errors by applying coding to the bits being transmitted.

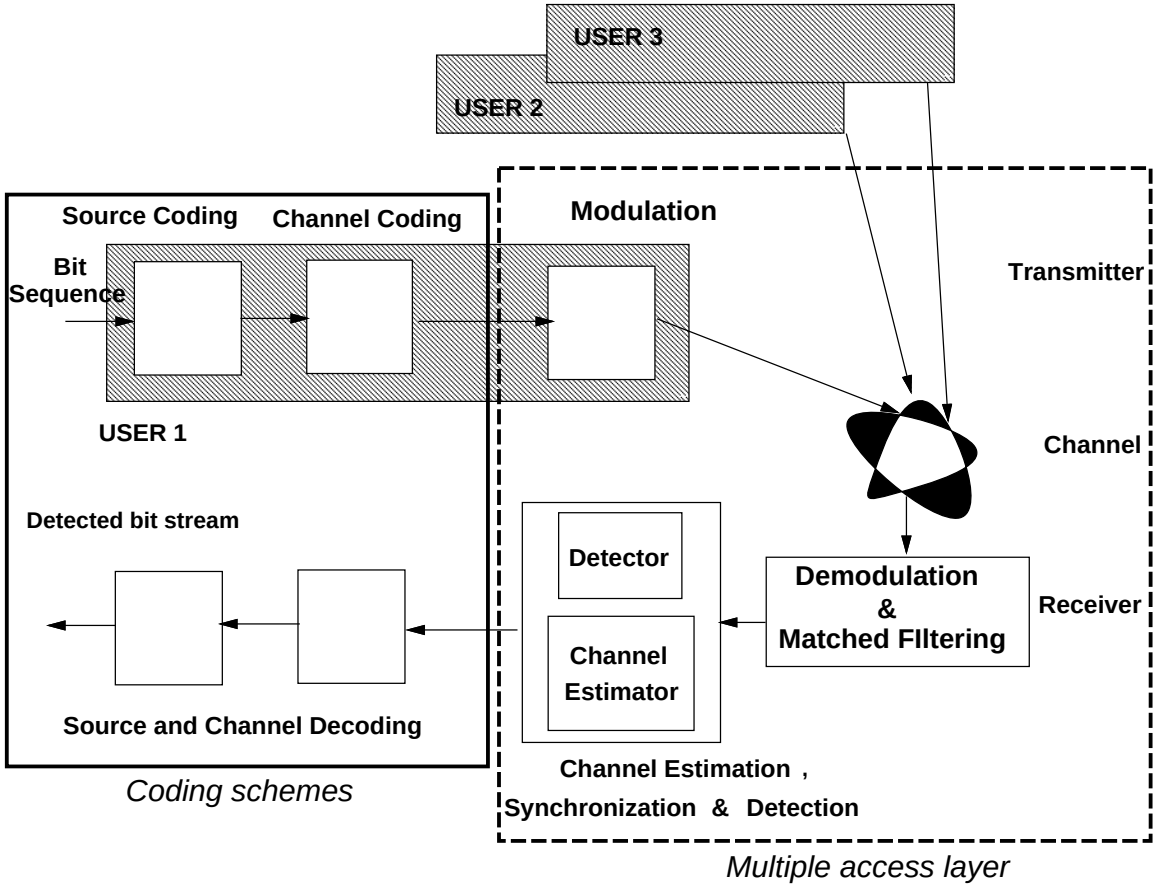


Figure 2.1 : A generic wireless link

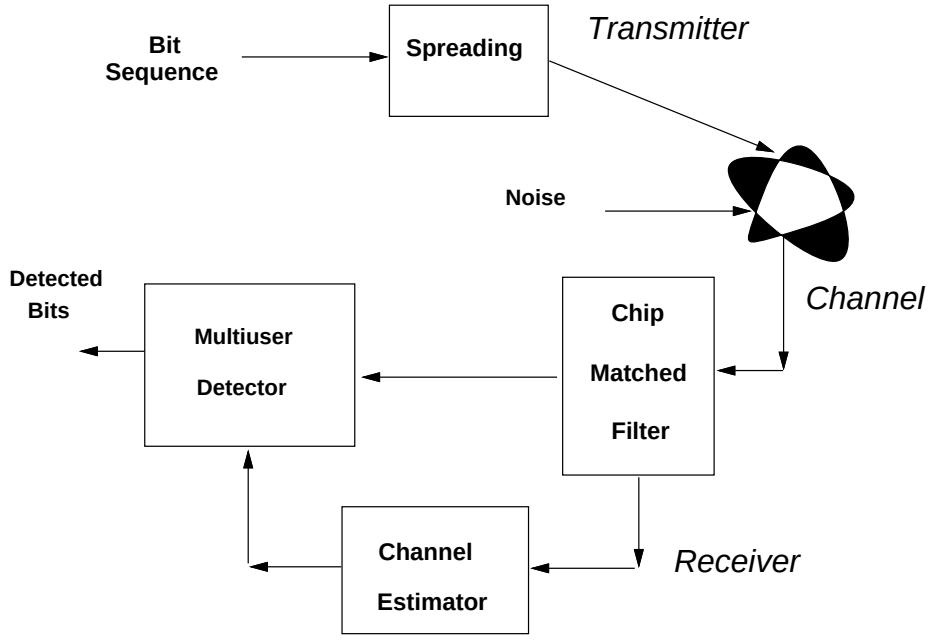


Figure 2.2 : The physical layer in the CDMA scheme

3. **Multiple access scheme:** A multiple access scheme is used to send the data from different users across the same channel. The physical layer (dotted portion in Figure 2.1), blown up in Figure 2.2, shows the Direct Sequence Code Division Multiple Access (DS-CDMA) scheme. Here each bit is "spread" into N *chips* as shown in Figure 2.3; where N is called the spreading gain. For example, if a bit is 1 μ second, and there are 7 *chips* in a bit, then the duration of each *chip* is $1/7$ μ second. Each user has a different spreading sequence (usually designed to be orthogonal to other such sequences).

4. **Modulation:** The *chips* are modulated by the RF carrier using a digital modulation scheme like binary phase shift keying (bpsk). The intermediate signal, composed of the superimposed *chip* streams of different users, entering the channel is called the baseband signal.

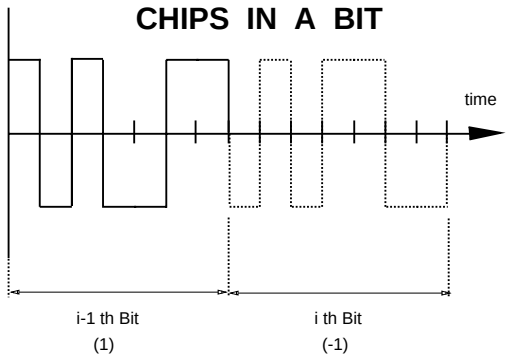


Figure 2.3 : Chipping or spreading sequence

The Channel: Several such transmitters send their modulated chip streams in the same frequency band and possibly at all instants. The transmission is a totally asynchronous process since users can start transmitting their sequences at any instant. These signals are superimposed in the channel and attenuated in strength. The channel also introduces noise, commonly modeled as Additive White Gaussian Noise (AWGN). Further, the composite signal from the channel results from multiple signals bouncing off obstacles, suffering varying delays and attenuations and getting superimposed. This is called the *multi-path* effect. The channel also introduces *fading*, which means that the attenuation of the different paths varies with time. Fading results from the relative motion of the transmitter and the receiver (Doppler effect) or the movement of the reflectors in the path of the radio signals.

The blocks on the **receiver** side are

1. **Demodulator and Chip Matched Filter:** This is the front-end of the receiver and it retrieves the baseband signal (composed of the superimposed chip-streams and noise) from the radio signal coming from the channel, for further processing. This continuous time baseband signal from the channel is converted

to a discrete time signal by sampling the output of a filter matched to the chip waveform. The chip matched filtering is an integration and dump of the baseband signal correlated with the chip waveform, over the chip duration, with the output held at the end of the chip duration.

2. **Channel Parameter Estimator:** Due to the asynchronous nature of the transmissions and due to the path delay introduced by the channel, the baseband signal corresponding to the bit-streams of different users is received in the receiver at different delayed instants in time. The multiuser detector needs the knowledge of the bit boundaries of all the users for detection. For interference suppression, certain kinds of detectors also need the amplitude and phase information of each user. The channel estimation procedure gives the arbitrary delay, the magnitude and phase change introduced by the channel for all the users. The initial part of the channel estimation involves an acquisition phase, where the first estimate of the parameters is obtained. For the rest of the transmission, it involves tracking, where the channel parameters are continuosly updated. Tracking is required for a dynamic channel, where the properties of the channel change with time due to multipath fading effects.

3. **Detector:** The detection process retrieves the bits of a particular user from the superimposed baseband signal, which has the chip streams of different users. The detector needs the knowledge of the spreading sequence of a user to detect the bits. The base-station receiver has a multi-user detection scheme [7], where the bits of all the users are detected. The detection problem involves solving the following equation(2.1) for the original bits $\mathbf{b}_i$, given the received information vector $\mathbf{r}_i$, the matrix $\mathbf{A}$ and $\mathbf{W}$ (obtained from channel estimation process) and

assuming AWGN Noise ν_i.

$$\mathbf{r}_i = \mathbf{AWb}_i + \nu_i. \tag{2.1}$$

With $\nu_i \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ and where $\mathbf{r}_i$ is a discrete observation vector formed at bit i by sampling the received signal and collecting N_c chip matched filter outputs; $\mathbf{A}$ is the $N_c \times 2K_{user}$ matrix of signal vectors, which depends on spreading codes and delays of each of the users; $\mathbf{W}$ is a $2K_{user} \times 2K_{user}$ diagonal matrix of complex amplitudes; $\mathbf{b}_i$ is the $2K_{user} \times 1$ vector of the K_{user} users' previous and current data bits; and the $N_c \times 1$ vector $\mathbf{K}$ is the noise covariance; N_c is the spreading gain and K_{user} the total number of users.

The chip matched filter output vector $\mathbf{r}_i$ is processed by a filter matched to the spreading sequence of a user to detect the bits $\widehat{\mathbf{d}}_\mathbf{i}$ (equation 2.2).

$$\widehat{\mathbf{d}}_\mathbf{i} = \mathbf{y}_i = \mathbf{S}^T\mathbf{r}_i \tag{2.2}$$

where $\mathbf{y}_i$ is the matched filter output and hence the initial detector output $\widehat{\mathbf{d}}_\mathbf{i}$; $\mathbf{S}$ is the N_c by K_{user} spreading matrix (N_c wide spreading codes for K_{user} users).

4. **Source and Channel Decoding:** These blocks are the receiver counterparts of the source and channel coding blocks on the transmitter side. The channel decoding extracts the bit stream before coding and detects and corrects some errors based on the property of the codes used. The source decoder uncompresses the bits from the channel decoder. This corresponds to the original bit stream sent across the transmission system.

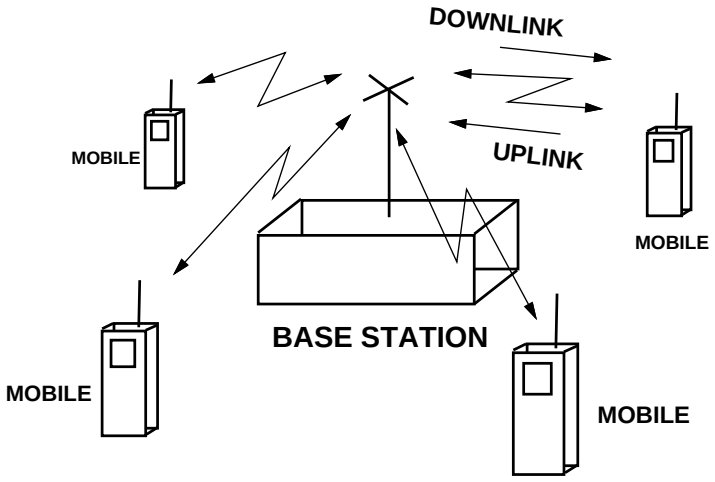


Figure 2.4 : Uplink and downlink

2.1.2 Uplink and downlink channels and system configurations

There are two types of wireless links based on the direction of transmission. The **uplink** is the link from the mobile (cellphone) transmitter to the base-station receiver and the **downlink** is the link from the base-station transmitter to the mobile-cellphone receiver (Figure 2.4). There are differences in the way the uplink and the downlink are modeled, both in terms of channel properties and the algorithms used. The multi-user detection and synchronization algorithms in the base-station receiver have a knowledge of the spreading sequences of all the users. The algorithms used in the mobile receiver are typically "blind" (i.e. they can work without the need for the knowledge of the spreading sequences of the other users). The blind schemes are also required to be computationally less demanding due to the limited processing power in the mobile unit.

2.2 System simulation of CDMA wireless networks

A complete system needs to have in one simulation setup all the components described previously. Further, the uplink and downlink have different components and configurations.

2.2.1 Requirements for simulating a CDMA system

The characteristics of CDMA systems call for the following special requirements for a simulation tool [8]:

- Fast and efficient data processing :

  The wide bandwidth generally associated with spread-spectrum systems requires large sampling rates in simulations. System-level simulations of these communication links normally employ a sampled-data representation of analog signals. This leads to long run times due to the large volume of data flowing between the various model blocks in a simulation. So it is required to move and process data efficiently.

- Multirate processing :

  The data rates in a CDMA system vary widely at various points. For example, the bit rate after the channel coding block increases by a factor which depends on the code used; similarly the data rate increases N (the spreading gain) times after the spreading block. Hence the system should be capable of accommodating these multiple rates.

2.2.2 State-of-the-art work in CDMA wireless communications and the need for an alternative method of simulation

Members of the Center for Multimedia Communications at Rice University develop new algorithms for the different blocks described earlier in this chapter. MATLAB is usually used for algorithm evaluation in this research. MATLAB is a programming language environment which has library functions for communication and signal processing applications. The language is similar to "C" in syntax and is interpreted by a simulation engine.

Further, simulation studies are mostly local to the block being considered. For example, Monte-Carlo simulations for evaluating a detection algorithm usually have a data source block, a detector block and some mechanism to count errors. The data source block is developed from some analytical expressions to give the effect of many users sending their data through the channel. While this is very useful for quick algorithmic exploration, it may not reflect the exact behaviour of the block in a real system.

A block diagram type simulator would be more convenient than MATLAB to build and simulate communication systems. Such tools have an environment where blocks corresponding to different algorithms are available in a library and can be connected to form a system of the type discussed before. Since many of the blocks were available as MATLAB code, Simulink was the primary choice for such a tool. Simulink uses the MATLAB engine for simulations and has a substantial library of functions for communication systems. The capabilities of such tools are discussed in the next section.

2.3 Block diagram type simulation and modeling environments

The block diagram based tools provide an interactive environment for the evaluation of communication systems. A modular system can be designed using blocks from built-in libraries or using custom blocks created from back-end MATLAB/C code. They can be classified into two categories according to the method of simulation (Figure 2.5):

- Cycle-based simulation :

  These tools simulate all blocks in the system for each time step and can be expected to provide comprehensive and reliable simulation results. A common clock provides global timing information to all the blocks.

- Dataflow or Stream-driven simulation :

  The simulation in these tools is triggered by the data entering a block. This method of simulation is computationally less demanding and well suited for signal processing applications.

The commercial tools such as Simulink, SPW, and COSSAP use a combination of these types of simulation [9–11]. These software tools also provide support for prototyping of designs by generating either C-code for Digital Signal Processors (DSPs) or VHDL code for Application Specific Integrated Circuits (ASICs).

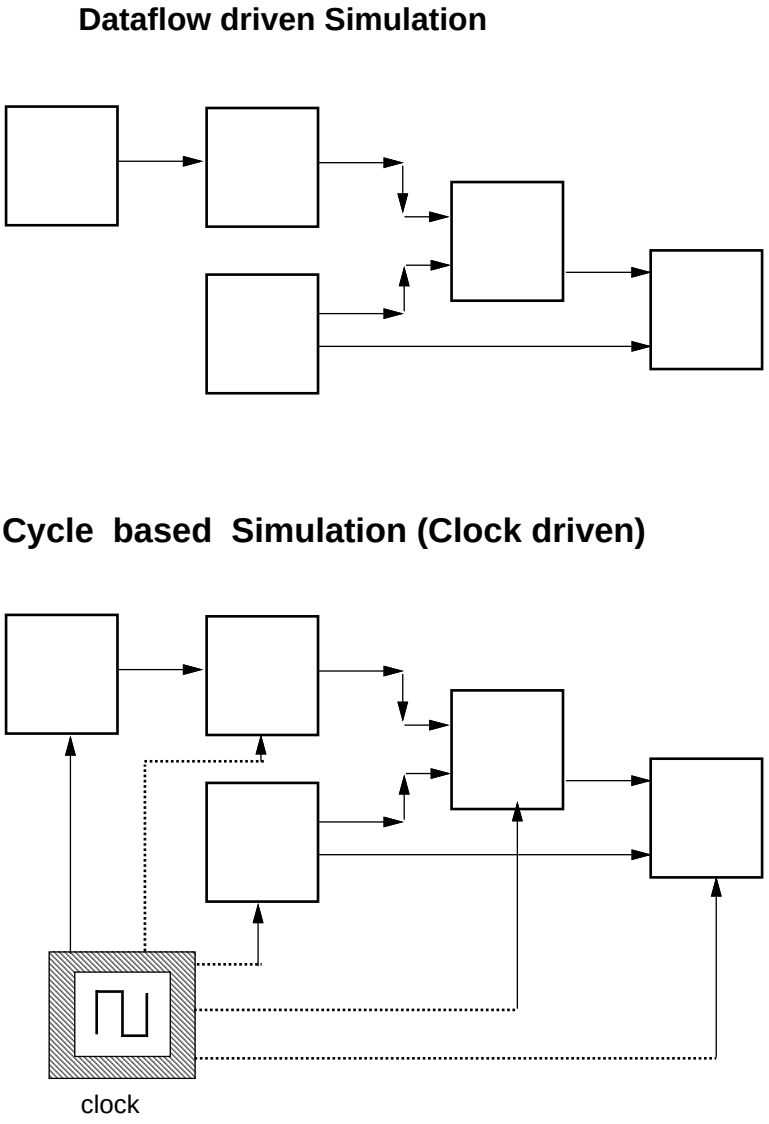


Figure 2.5 : Dataflow driven simulation vs. cycle-based simulation

Chapter 3

Simulation System for CDMA Wireless Links

This chapter details the construction of the software simulation system implemented around a physical layer multiuser CDMA wireless link shown in Figure 3.1. This link is the backbone of the software testbed to which other modules are added.

3.1 Construction of the Simulink based system

The simulation system is built as a Simulink block diagram [12]. The main blocks of the system correspond to blocks in the prototype CDMA communication system introduced in chapter 2 (Figure 2.1). The user data block generates the bit sequences corresponding to each user, spreads them according to the spreading code, and conveys the spread signals to the channel block where the chip sequences of all the users are superimposed and Additive White Gaussian Noise (AWGN) is added to it. The output from the channel is a baseband waveform discretized at the system sampling instants. The baseband waveform is processed in a filter matched to the chip waveform. The mathematical analysis up to this part is presented in the appendix A.2.1. The chip matched filter block generates an output for every chip duration, which is then vectorized and fed into the channel estimation block.

Maximum likelihood (ML) channel estimation uses a fixed preamble which is known at the receiver [13, 14]. The preamble bits are sent across the channel in the initial part of a transmission. The ML channel estimation block uses the result-

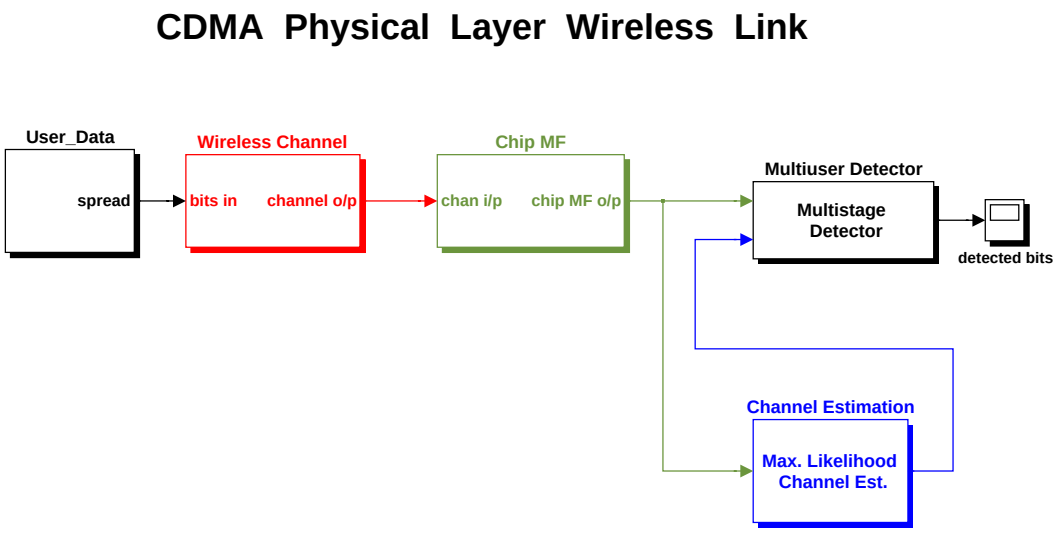


Figure 3.1 : The basic multiuser CDMA link in Simulink

ing output from the channel in the form of the chip matched filter vector and the knowledge of the preamble bits to estimate the channel parameters (amplitude, and delay information). The multiuser detector uses the chip matched filter outputs and the matrix which conveys the delay and amplitude information to detect the bits of all the users. The analysis is presented in the appendix (section A.2.2).

3.1.1 Features of the multiuser physical layer link

- **Spreading Methods:**

  The system uses a variety of spreading codes with Binary Phase Shift Keying (BPSK) as the chip waveform. It uses Gold codes [15], M-sequences [16] and random spreading codes. Gold codes and M-sequences are a set of spreading sequences which have the property of very low cross-correlation. They are pseudo-random sequences which can be implemented as a feedback shift register. The implementation in the software testbed uses a look-up table method for

the spreading. The random spreading codes are set up at the beginning of a simulation by choosing a random set of binary sequences for all the users. The construction of the spreading block is further dealt with in the appendix (B.1.2).

- **Channel model:**

  The software testbed uses a static Additive White Gaussian Noise (AWGN) multipath channel, the most commonly used model for a wireless channel. It assumes that the channel superimposes all the signals and adds white noise(noise power is uniformly distributed over all frequencies). The multipath effect is due to a superposition of signals from different paths between the transmitter and the receiver. The data passing across the channel is in the form of a discrete signal (with a finite number of samples per chip). The channel output is available at the sampling rate of the simulation system (four times every chip duration in the basic version). The discretized channel output is passed through a chip matched filter matched to the chip waveform (a rectangular pulse). The modeling of the channel and the following chip matched filtering is presented in the appendix(B.1.2).

- **Maximum Likelihood Channel Estimation:**

  The maximum likelihood method is used for channel estimation [13, 14, 17–19]. A preamble $X_{preamble}$ of length $L_{preamble}$ is sent initially in a simulation with a sequence of L_{total} bits. The corresponding output of the channel processed in the chip matched filter is observed as Y_{cmf}. Y_{cmf} and $X_{preamble}$ are used to estimate the matrix $\mathcal{U}\mathbf{Z}$ which captures the effect of all the channel parameters. Further details are presented in the appendix (section A.2.2). The matrix $\mathcal{U}\mathbf{Z}$ can be

used to calculate the code matched filter output $\mathbf{y}_i$ and the cross-correlation matrix $\mathbf{R}$:

$$\mathbf{y}_i = (\mathcal{U}\mathbf{Z})^T \mathbf{r}_i \tag{3.1}$$

where $\mathbf{r}_i$ is the chip matched filter output vector; and

$$\mathbf{R} = (\mathcal{U}\mathbf{Z})^T (\mathcal{U}\mathbf{Z}). \tag{3.2}$$

- **Multiuser detection:** The software testbed has a choice of three detection schemes: the conventional matched filter detector and two multiuser schemes.

  1. **Matched Filter:** The basic detector is a matched filter detector, which correlates the received waveform with the suitably delayed version of the spreading code. It does not cancel the effect of interference from other users. The equation 3.3 gives the relation with the original bits.

$$\widehat{\mathbf{d}} = \mathbf{y} = \mathbf{RAd} + \nu \tag{3.3}$$

     where $\mathbf{y}$ is the K chip matched filter vector; $\mathbf{R}$ is the K by K correlation matrix; $\mathbf{A}$ is the K by K amplitude matrix; $\mathbf{d}$ is the K vector of data bits and ν is the noise vector.

  2. **Decorrelator:** The decorrelator is a linear detector [7], which applies a linear transformation to the matched filter output to reduce the effect of multiple access interference(MAI). The transformation $\mathbf{R}^{-1}$ is applied in a

decorrelator which eliminates MAI (where $\mathbf{R}$ is obtained from the channel estimation process). So

$$\widehat{\mathbf{d}}_{decorr} = \mathbf{R}^{-1}\mathbf{y} = \mathbf{A}\mathbf{d} + \mathbf{R}^{-1}\nu. \tag{3.4}$$

3. **Differencing Multistage:** This is an efficient version of multistage detectors which uses a parallel interference cancellation technique [20]. Here the detector estimates and cancels the interference MAI for each user in parallel. This method removes the necessity to calculate the inverse of $\mathbf{R}$, by using an iterative method to successively cancel the interference. In multistage detection, the l^{th} stage of the detection is

$$\widehat{\mathbf{z}}^{(l)}_{multistage} = \mathbf{y} - \mathbf{R}\mathbf{A}\widehat{\mathbf{d}}^{(l-1)} \tag{3.5}$$

with

$$\widehat{\mathbf{d}}^{(l-1)} = sign(\widehat{\mathbf{z}}^{(l-1)}_{multistage}) \tag{3.6}$$

where $\widehat{\mathbf{z}}^{(l)}_{multistage}$ is the soft decision estimate in the l^{th} iteration and $\widehat{d}^{(0)} = sign(y)$.

The differencing multistage detector calculates the differential between 2 stages l and $l-1$ instead of the soft decision in the l^{th} stage to save computations [21].

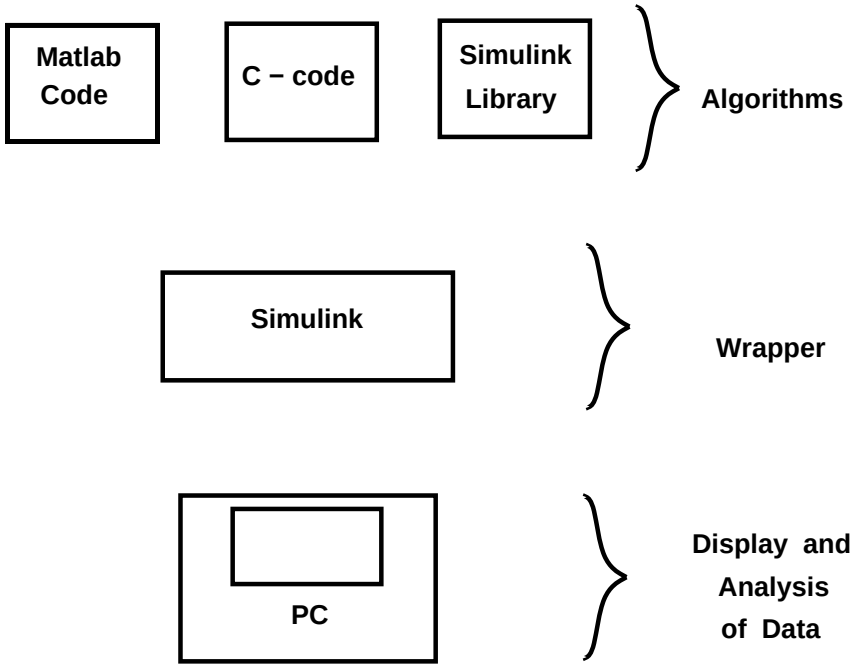


Figure 3.2 : The utilization of different libraries

3.1.2 System development using Simulink

The methodology of development of the simulation system is presented here. The different components for the system are obtained from different sources/libraries as shown in Figure 3.2. A Simulink block diagram can use blocks from the standard libraries or from models defined using m-file S-functions (Simulink-functions) or C-MEX S-functions. A major factor in the choice of the Simulink system was its usefulness as a *wrapper* for available code. The Simulink S-functions are modifications of legacy code used for earlier evaluations. This method has the advantage of quick development.

3.1.3 Global management of simulation time-steps in Simulink

The simulink system [12] uses a combination of the simulation methods described in chapter 2 (section 2.3). There is a discrete time-step (simulation clock) based on the fastest sampling rate. However, different blocks process data at different rates depending on the rate at which inputs are available.

As mentioned earlier, a Simulink block diagram is built using components from the standard libraries and from the blocks built as S-functions [22]. The S-functions are a format for defining a block. The complete behavioral description of the block is specified by these S-functions. The S-functions have modules for initializing the sizes of the ports, setting the initial conditions, defining the derivatives for a continuous system, finding the next state for a discrete system, and generating the output for a set of inputs in a particular state.

The Matlab simulation engine fires the proper sequence of blocks in the order of data-flow. The blocks on the source side of the signal-stream execute first; and on the availability of the outputs of these blocks, the blocks which consume their outputs execute.

3.2 Features of the simulation system

This section presents the important features of the simulation environment. The final system with a multiuser physical layer link is shown in Figure 3.3.

3.2.1 Modular simulation system with a DS-CDMA library

The system is modular in design and allows easy addition of blocks. A core library of components was created to flexibly model the CDMA wireless link. Simulink

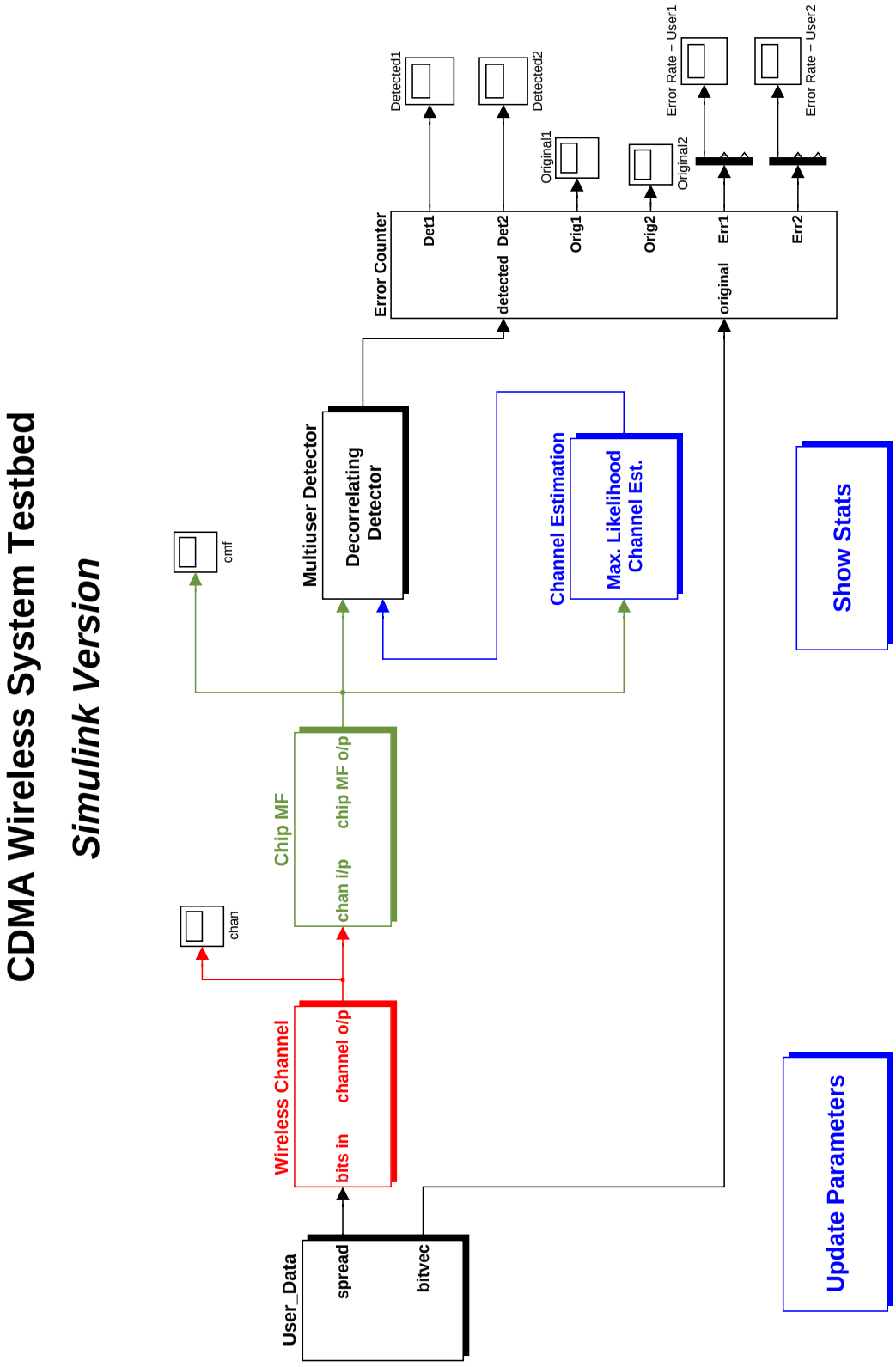


Figure 3.3 : The simulink version of the software CDMA testbed

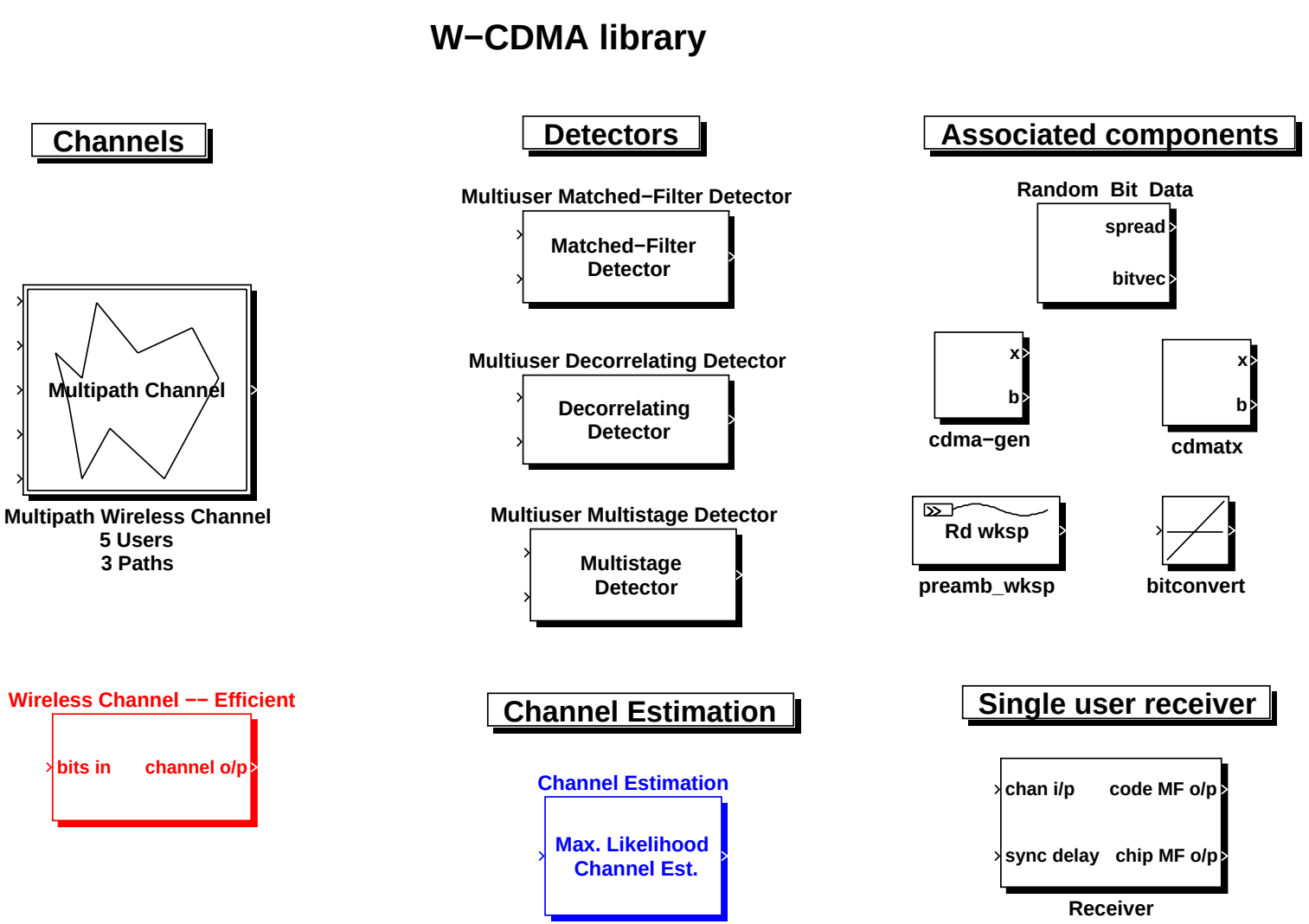


Figure 3.4 : The simulink CDMA library

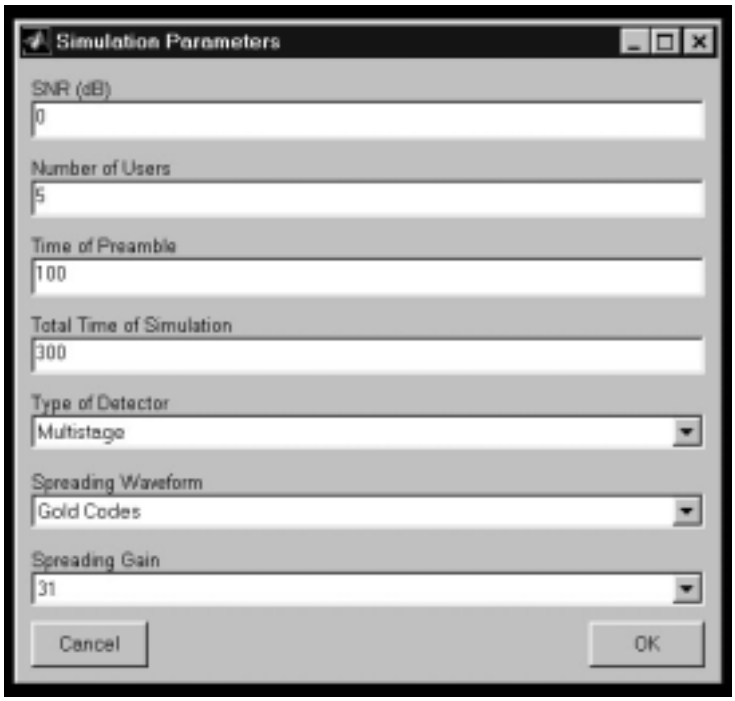


Figure 3.5 : The graphical user interface

has proprietary communication and DSP libraries. While components from these libraries are extensively used, many of the required blocks for CDMA systems are not yet available here. The library (Figure 3.4) is comprised of the major components of the physical layer link, described in the previous section.

3.2.2 Flexible system

The simulation testbed is a flexible system which can be reconfigured using a graphical user interface (GUI) (Figure 3.5). The GUI was developed using Matlab handle graphics. This interface allows the variation of system parameters. The GUI can be used to set the workspace variables corresponding to parameters such as number of users (K), spreading gain (N), number of bits in the stream ($L_{preamble}$ and L_{total})

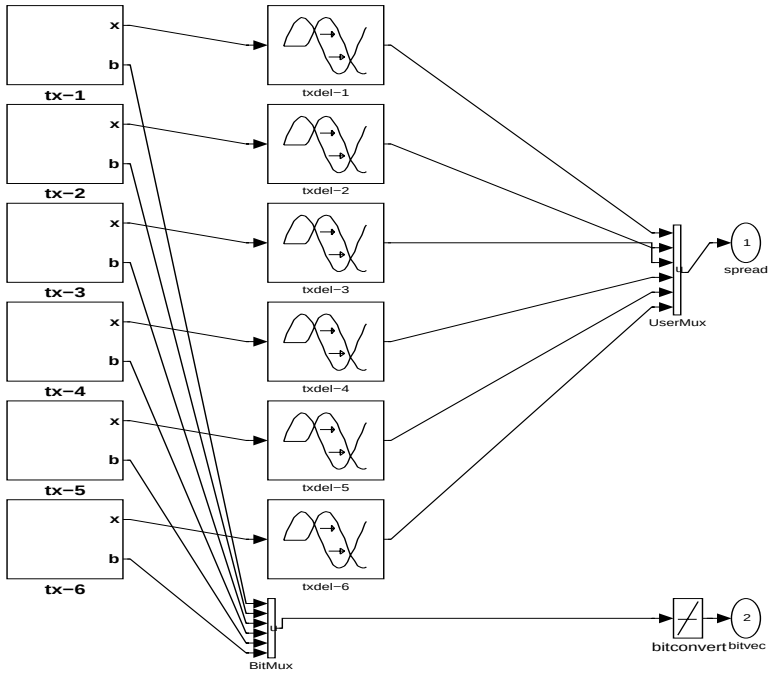


Figure 3.6 : Automated netlist generation

and number of paths (P). The GUI also allows the reconfiguration of the system by interchanging the modules (as in multiuser detection and spreading methods).

3.2.3 Automated net-list generation

The system uses an innovative method of automated block diagram creation, which allows the configuration of the system with simulation parameters. This is described in more detail in the appendix (section B.1.2). For example, when the number of users is set to 6, the data generation block is reconfigured as shown in Figure 3.6.

3.3 Capabilities of the system

The system is capable of studying typical scenarios such as Uplink, Downlink and a phone to phone call, all of which occur in a CDMA system. Figure 3.7 shows an

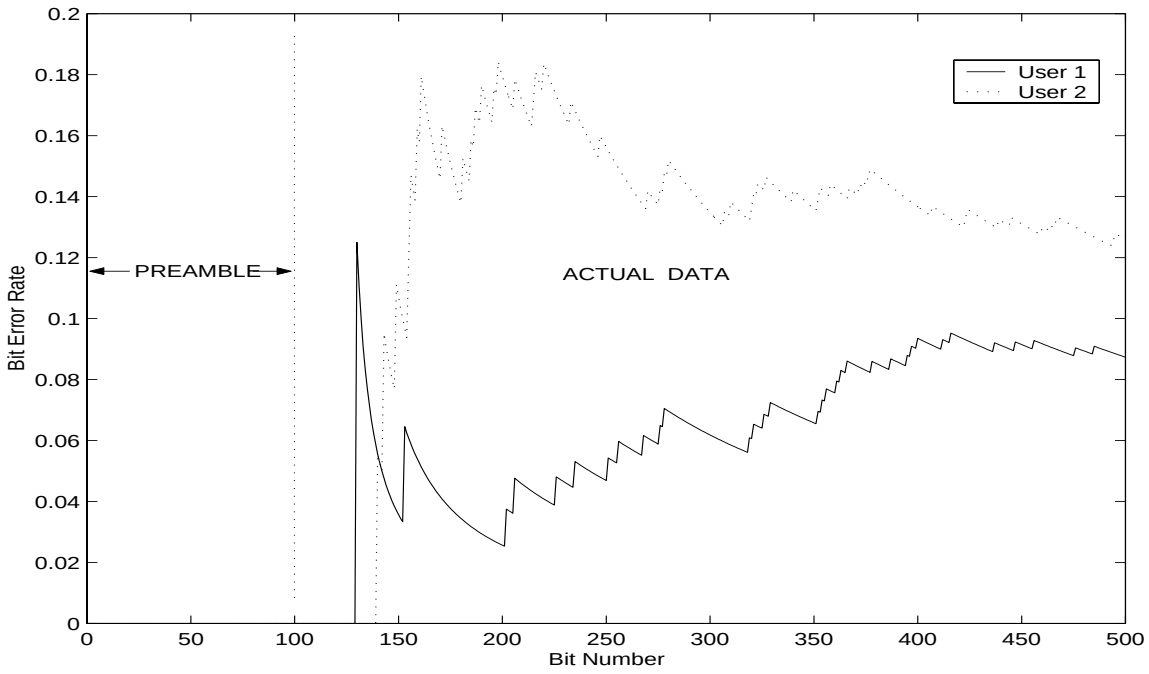


Figure 3.7 : An example run showing the bit error rate comparison of 2 users

example run for an uplink system with 10 users, an SNR of 0 dB, equal signal power for all users and single path. The first 100 bits comprise the preamble used in the estimation of channel parameters and the following 400 bits are the bits conveying useful information. The real-time plots can be used to compare the bit error rates (BERs) of different users (users 1 and 2 in this case).

The system can be modified to simulate a complete system like the one shown in Figure 3.8, which shows a one-way call from one mobile phone to another. This has both the uplink and the downlink systems in place. The real-time behavior of such a block diagram is expected to be close to that of a real life system. Further, such a simulation scheme is scalable to include more mobile phones and base-stations in the simulations.

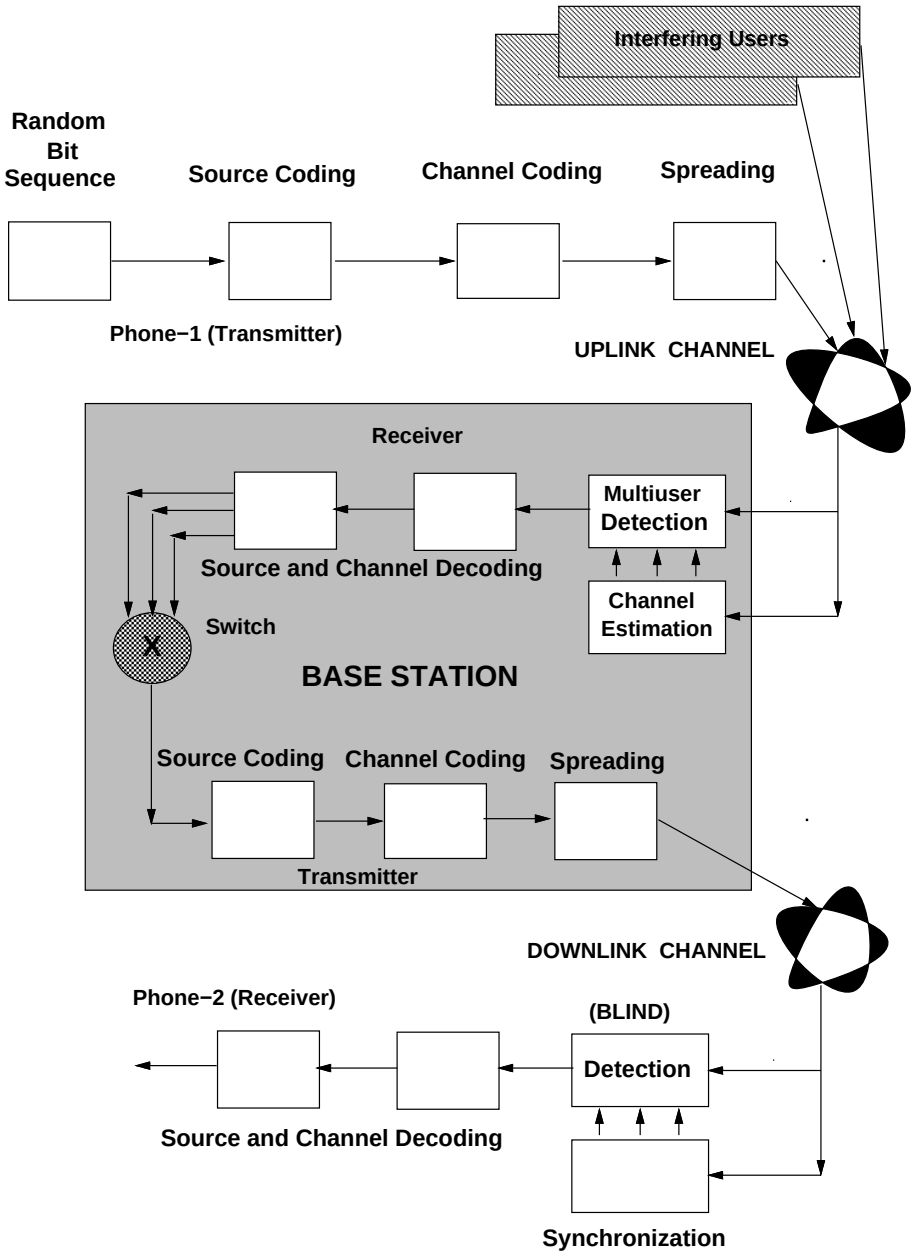


Figure 3.8 : A phone call in a CDMA wireless network

Chapter 4

An Efficient Method of Simulation

This chapter describes an efficient method of simulation which provides accurate simulation even while keeping the sampling rate lower than that in the traditional method of modeling such a system. The earlier method of simulation described in the previous chapter explicitly represents the baseband signal passing through the channel as a discrete time waveform (sampled many times a chip duration). This form of modeling not only tends to be slow in simulation speed but also introduces an error due to the representation of the continuous time baseband signals as discrete signals.

The new method combines the spreading, channel and the chip-matched filtering in one block, which obviates the need for sampling the channel at the sub-chip level. In this method the intermediate signals are not explicitly represented. This not only speeds up the simulation but also gives a more accurate value for the chip-matched filter output, which is free from the defects of sampling. This distinction is made clear in Figure 4.1.

A part of this study uses a quantized version of the delays of individual users' bit stream to evaluate the chip-matched filter output vector using the efficient method. The usage of the different version of the delays captures the effect (and defects) of sampled representation of channel signal in the simulations.

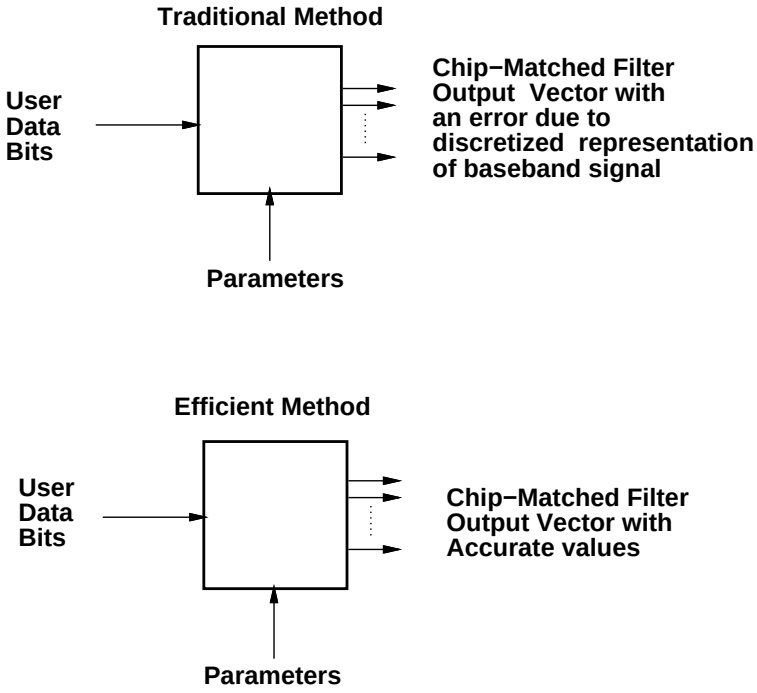


Figure 4.1 : The two methods of modeling – The big picture

4.1 Analysis of the chip-matched filtering

Figure 4.2 shows the part of a real system being considered. The bit-streams of different users are spread using the respective spreading codes and the resulting chip streams are superimposed and fed into the channel. The addition of noise results in the baseband signal. This continuous time baseband signal at the output of the channel is converted to a discrete time signal by sampling the output of a filter matched to the chip waveform.

However, a simulation system like Simulink models all the signals as discrete time signals. Therefore even the baseband signal through the channel has a discrete time representation. Hence the output of the chip-matched filter has to be derived from this discrete time channel output (which is available as a finite number of samples per

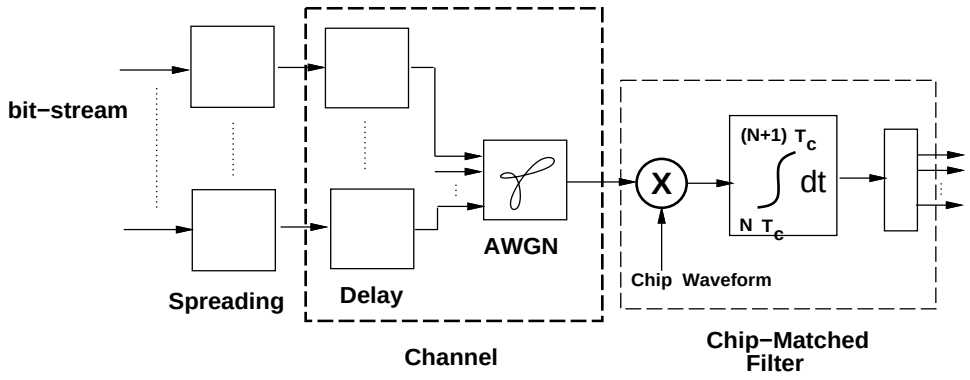


Figure 4.2 : The part of the system under consideration

chip). The use of a discrete channel output introduces an error into the chip-matched filter output, which can be abated only by increasing the channel sampling rate.

The Simulink block diagram in Figure 4.3 shows the traditional method of modeling this part of the system. A bit-stream is generated as a random sequence for each user and is spread using the spreading waveform of that user. The sequence of chips is delayed to model the asynchronous nature of the transmissions and the path delays in the channel. The spread signals of all the users are superimposed and fed into the channel. This system uses a minimum sampling time equal to a fraction of the chip duration. Consequently, the channel output is available at the sub-chip sampling rate (typically 4 times a chip).

Consider the output vector from the chip-matched filter (see appendix, section A.2.1). This vector is formed by grouping the chip-matched filter outputs from N consecutive chip durations. The chip-matched filtering is an integration over the chip duration, with the output dumped at the end of the chip duration. The value held before the dump is the chip-matched filter output. The most intuitive method of doing the integration is to calculate the area under the curve for the received signal (as

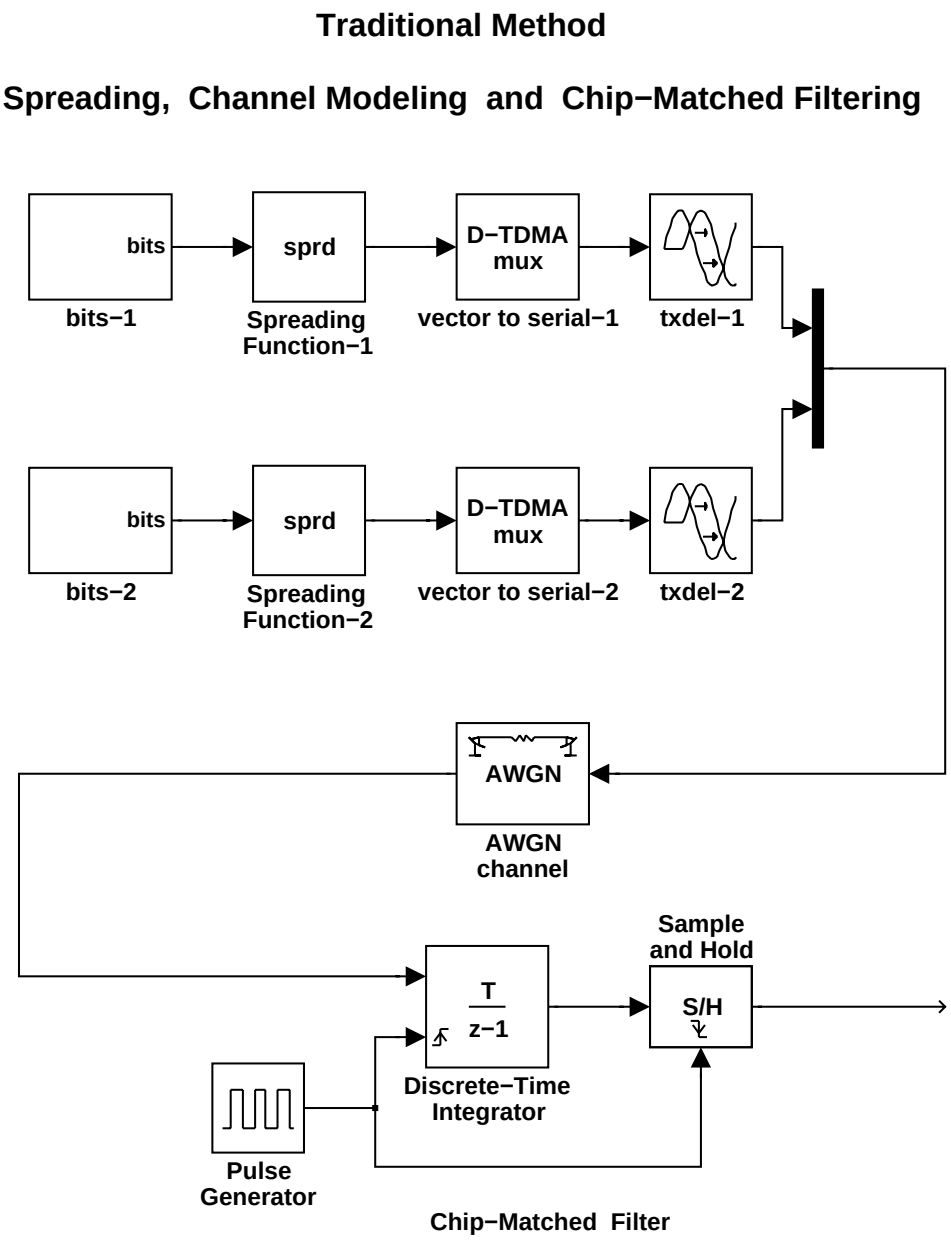


Figure 4.3 : The traditional method of modeling in Simulink (with sampled signals)

shown in Figure 4.4a). However, this method cannot be used in a sampled system like Simulink, where only a finite number of samples are available from the channel. The chip-matched filter outputs are generated by adding the areas between sampling instants. The successive sampled values are used to find the areas between two sampling instants, and all such areas in a chip duration are summed up. This approach (Figure 4.4b) is akin to a numerical integration as compared with the "area under the curve" method. This numerical integration method does not give an accurate value for the output of the chip-matched filter. The sampling needs to be of a finer granularity to reduce the defects in this method of data generation.

A comparison of the integration methods indicates that the numerical integration approach takes a comparatively longer computation time than the "area under the curve" method.

4.2 An efficient method of simulation modeling

We incorporate a novel method of modeling which combines the spreading, channel modeling and matched filtering in one block as shown in Figure 4.5. This method gives accurate results while keeping the system sampling time to once a chip.

Figure 4.8 shows the new method of modeling in Simulink. Here the separate bit streams are fed into a single block which models the spreading, channel modeling and the chip-matched filtering. Each user has a unique spreading code (Figure 4.6), which modulates the bits, and all these chip streams are superimposed in the channel. The superimposed chip stream over any bit period is essentially a linear combination of all the users' spreading codes which are delayed and suitably aligned with each other. This baseband waveform over a bit duration (N chips) depends on the 2 consecutive bits of all the users (Figure 4.7). This block uses the present and the previous bits

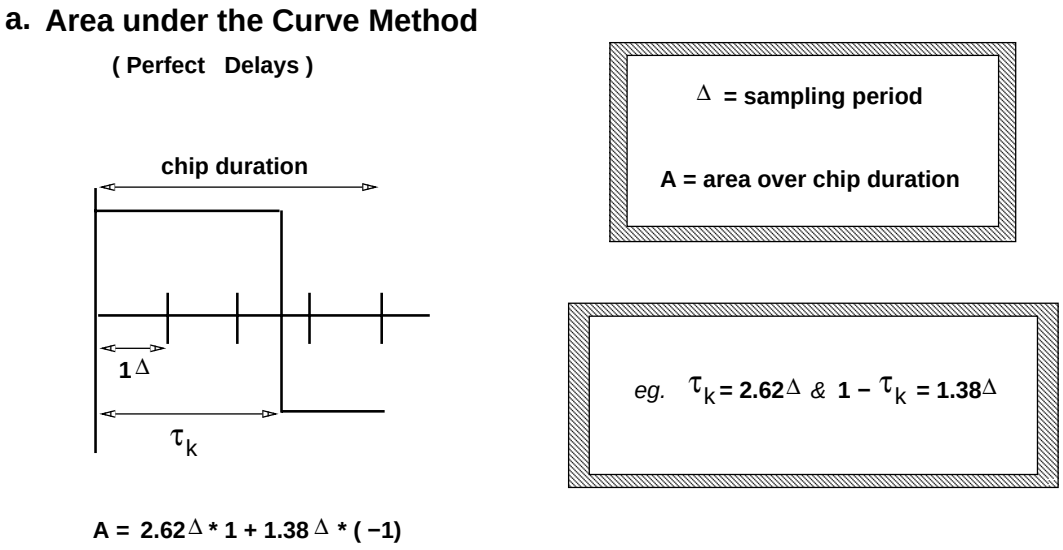


Figure 4.4 : The integration methods in the chip-matched filtering

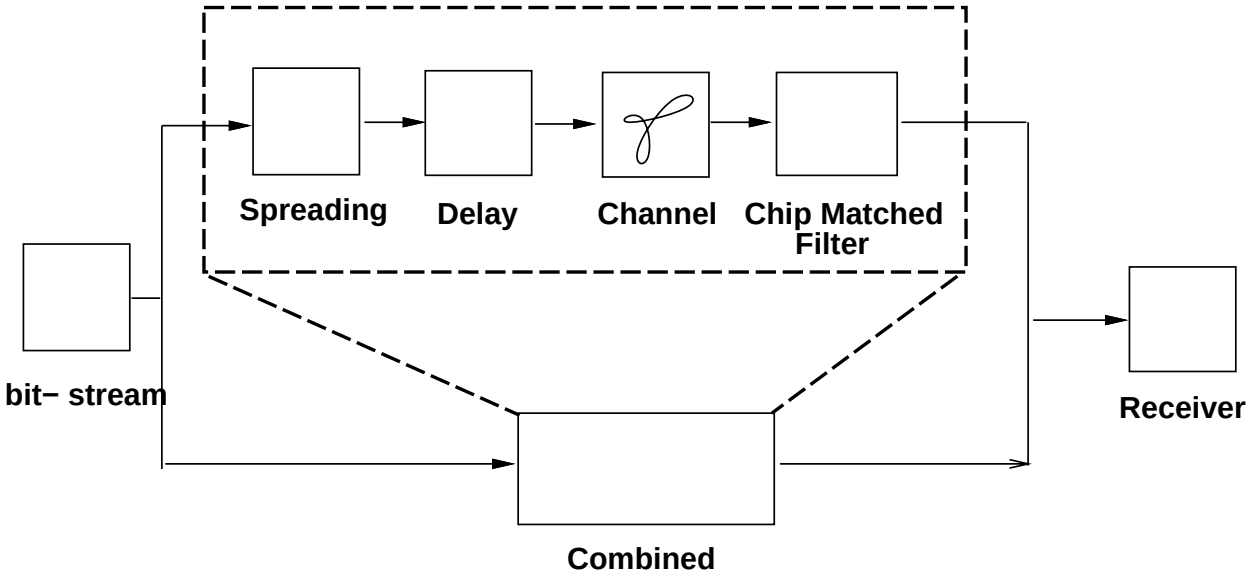


Figure 4.5 : Comparison of the two methods of modeling

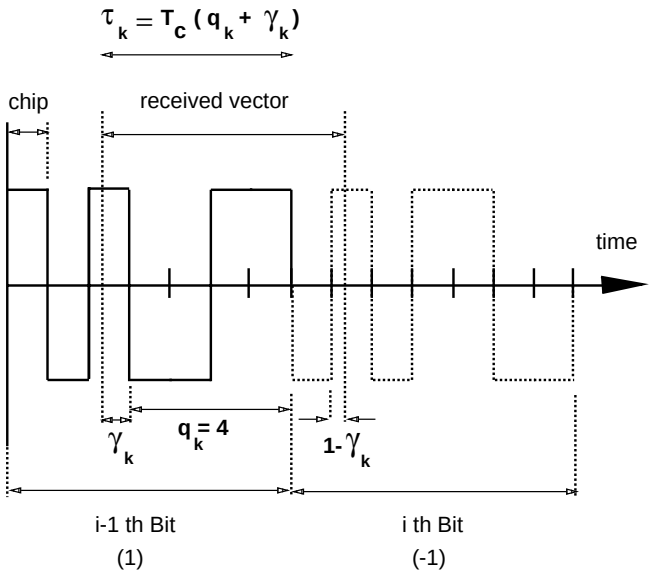


Figure 4.6 : Received signal – *chip* level details

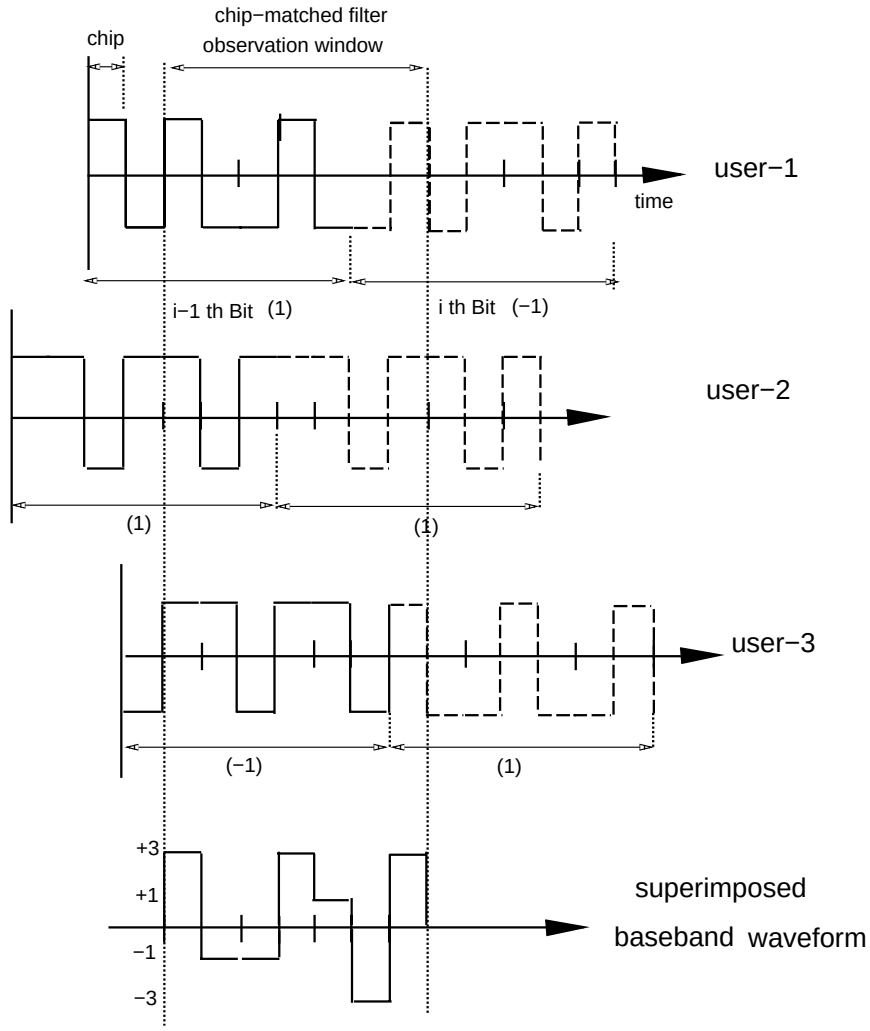


Figure 4.7 : Superimposition of chip streams of 3 users (with 7 chips per bit)

of all users, the individual delays, the amplitudes and the channel parameters to calculate the chip-matched filter output.

In this simulation system, there is no explicit representation of the intermediate signals such as chip sequences, noise and baseband channel waveform. A Matlab S-function (Simulink-function) is used to directly calculate the chip-matched filter outputs from the current and previous bits of all the users and the channel prop-

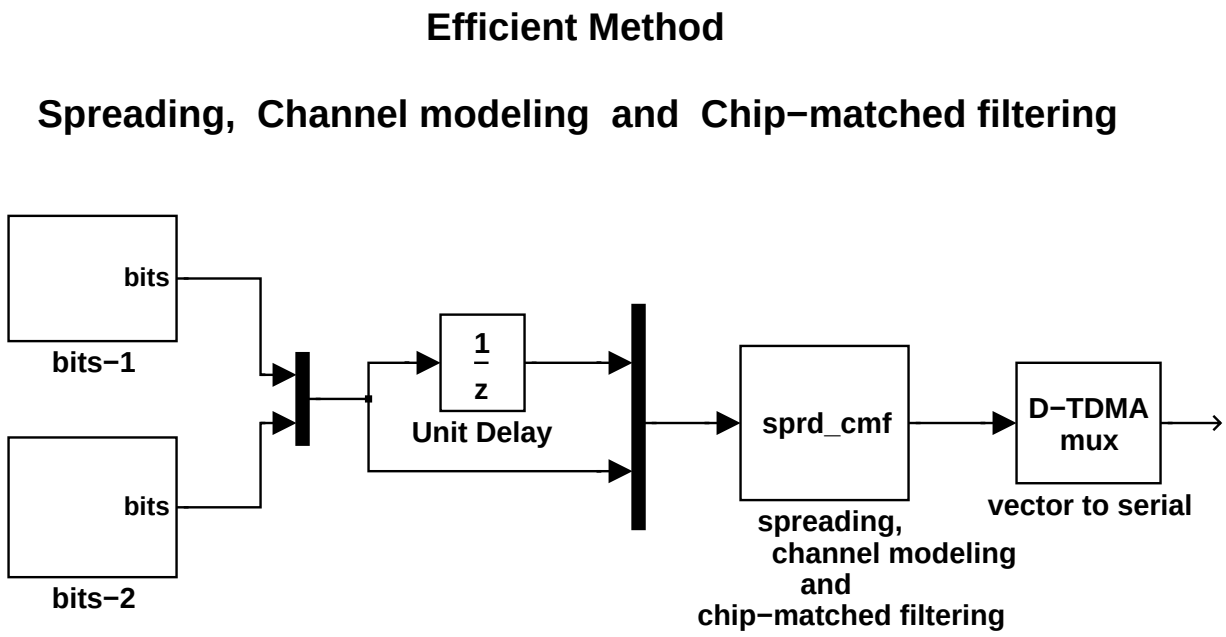


Figure 4.8 : The efficient method of modeling in Simulink

erties. This corresponds to the accurate "area under the curve" method, shown in Figure 4.4a.

4.3 Using the efficient method with quantized delays to capture the effect of sampling

This section presents a method which uses the efficient method of simulation with a different version of the delay parameters to model the effect of sampling.

The perfect delays, τ_ks, are quantized to the next sampling instant to obtain the quantized versions, $\tau_{k_{quant}}$s, as shown in Figure 4.4c. The $\tau_{k_{quant}}$s are used instead of the perfect τ_ks in the channel data generation. It is seen from simulation results that the chip-matched filter output vector $\mathbf{r}_i$ obtained here (using the "area under the

curve" method) is the same as that of the system with discrete time representation of signals. Hence the use of quantized versions of delays models the effect of sampling. This is so because even if the $\tau_{k_{quant}}$s were the delays instead of the τ_ks, the chip-matched filter output for a sampled system would have been the same. The analysis for this method is shown in Figure 4.9; a CDMA system with 4 samples per chip is considered with the sampling instants as shown. The example shows that the sampled value and the resulting chip-matched filter output are the same in the case where the perfect delays (τ_ks) are used and in the case where the quantized version of the delays ($\tau_{k_{quant}}$s) are used.

For example, if the perfect delay for the k^{th} user was 27.3429 units, then the quantized value would be 27.5 units. Let the sampling occur at instants $27.00 + \epsilon$, $27.25 + \epsilon$, $27.5 + \epsilon$, $28.0 + \epsilon$ etc., where the ϵ signifies an infinitesimal time difference between the end of one sampling period and the beginning of the next.

Hence by using a modified version of the delays for the parameters, this method of modeling captures the effect of sampling. It is argued that such an approach in simulating a CDMA system is computationally more efficient than the older method.

4.4 Comparison of simulation time for the two methods of modeling

In addition to evaluations using the Simulink block diagrams, a stand-alone Matlab program is used for profiling runs. The Matlab program takes in the different parameters and generates the channel data in the form of chip-matched filter outputs. The results from the stand-alone matlab simulations of the data generation part are presented first.

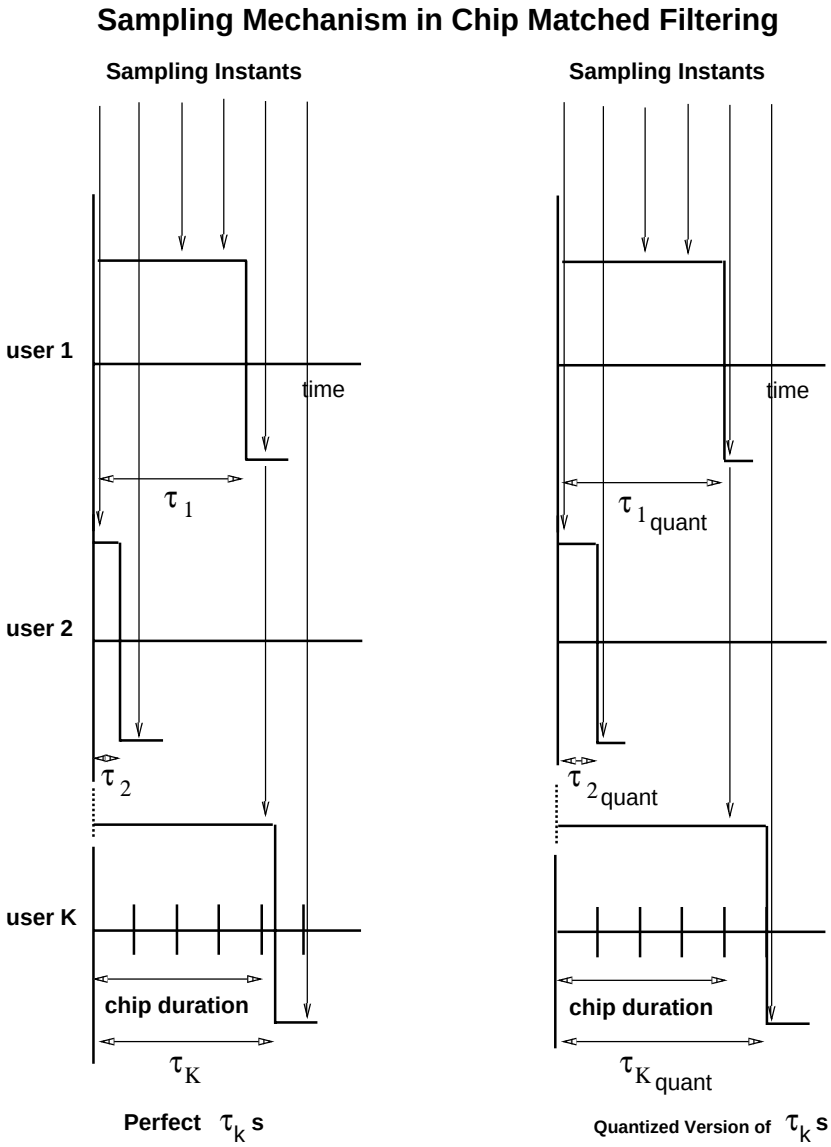


Figure 4.9 : Sampling mechanism in chip-matched filtering

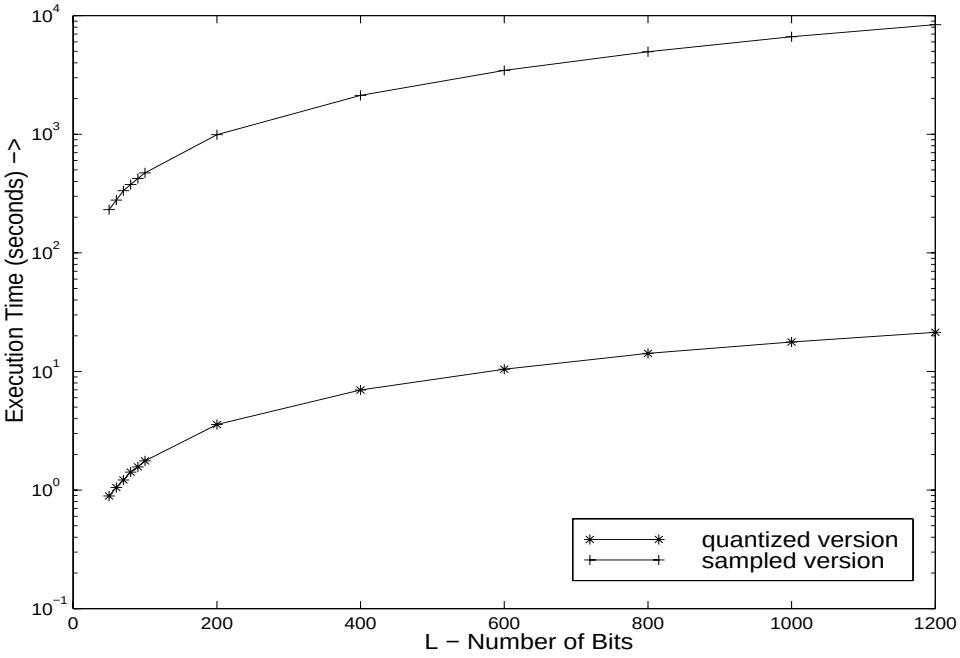


Figure 4.10 : Execution time plot for different bit-stream lengths: L $\in$ [50,1200]

4.4.1 Execution time comparison in the stand-alone Matlab simulation

The experiment evaluated the chip-matched filter output using both the methods as detailed in the previous sections. It was observed that the output vectors were the same in both methods for the same set of randomly generated data ($\mathbf{A}$, τ_k, $\mathbf{W}$, $\mathbf{b}_i$ & ν_i).

The execution time required to generate the chip-matched filter output for the two methods was computed using the `tic and toc` functions which return execution time in seconds. It is not a perfect measure as the execution time varies depending on the load on the workstation. The experiments were done in MATLAB and used a method which would ensure a fair comparison of the execution time.

In the first experiment, the number of bits in the stream was varied in the range

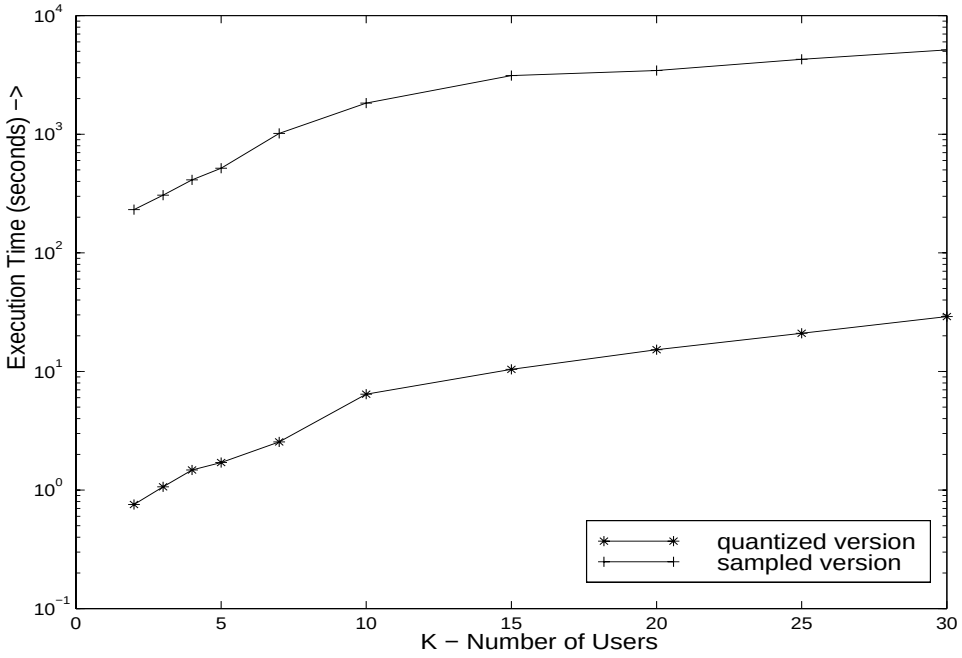


Figure 4.11 : Execution time with different number of users : K $\in$ [2,30]

$L = [50, 1200]$. The number of users K was 5, and a spreading code of length $N = 31$ was used with 8 samples per chip (Figure 4.10). The plots indicate that the area under the curve method with quantized delays takes 2 orders of magnitude less execution time than the numerical integration method (done iteratively). The next experiment was to vary the number of users in the range $K = [2, 30]$. Again a spreading code of $N = 31$ was used, with $L = 100$ and 8 samples per chip. This is shown in Figure 4.11. Both the above experiments indicate a significant improvement in the execution time for a wide range of system parameters.

4.4.2 Simulation time comparison for the entire Simulink system

The wall clock execution times were compared for different lengths of data for the two systems. The plots in Figure 4.12 show that the execution of the simulation testbed

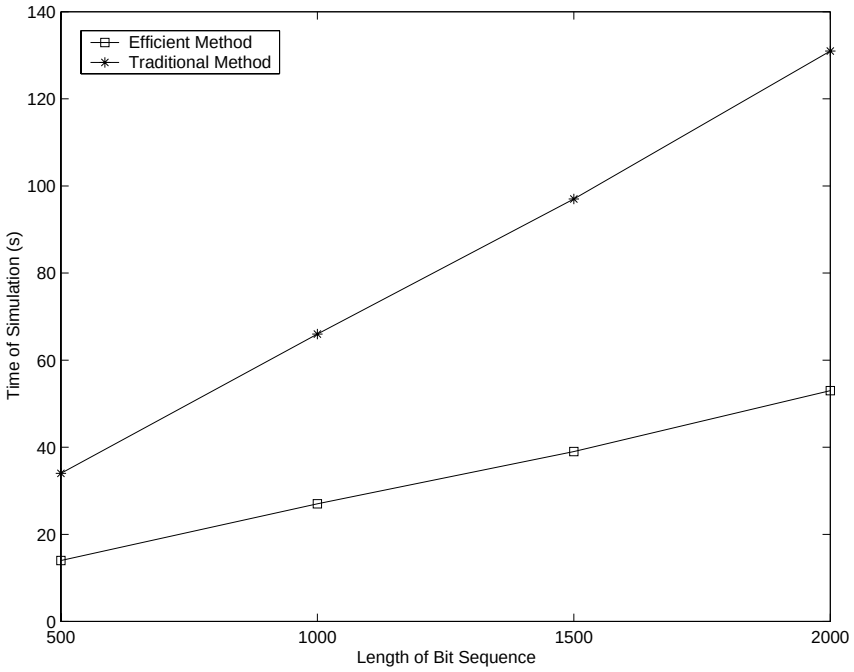


Figure 4.12 : Comparison of simulation time with different bit-stream lengths (5 users)

improved by a factor of about $2\frac{1}{2}$. The older method uses 4 samples a chip during the chip-matched filtering, and the efficient method reduces that to once a chip duration. The simulation time improves by a factor of less than 4 because the data generation method is more complicated.

4.4.3 Profiling analysis for the Simulink based system

This section presents the profiling analysis for the two methods of simulation modeling. The profiling method is described in the appendix (section B.2). The simulation time break-ups are shown in table for the two methods. The profiling data is presented for systems with two different multiuser detection schemes: decorrelating and multistage detectors, in tables 4.1 and 4.2 respectively. The time taken for the pulse generator function (which drives the simulation timesteps) is reduced by a substan-

tial amount in the efficient method of modeling. However, the combined spreading, channel modeling and chip-matched filtering take a larger percentage of the total simulation time due to the underlying code being more complicated. The synchronization and detection blocks take similar execution times for both the methods of simulation.

	Traditional method		Efficient method	
Component	Time (s)	Percentage	Time(s)	Percentage
Total	147	100	41.9	100
Simulation time-steps (pulse generation)	88.21	60	14.14	33.7
Spreading	9.48	6.4	-	-
Combined spreading, channel modeling and chip-matched filtering	-	-	16.56	39.5
Channel estimation	0.61	0.4	0.49	1.2
Decorrelating detector	1.75	1.2	1.74	4.1

Table 4.1 : **Comparison of profiling data for the two methods of simulation for the system with a decorrelating detector:** since the total time of simulation is different for the two cases, the percentages should be interpreted carefully. In the efficient method, there is a substantial reduction in the time required for setting up the simulation time-steps using a pulse generator. The block for combined spreading, channel modeling and chip-matched filtering takes more simulation time than the earlier spreading block, because of its complexity. The channel estimation and detection blocks take similar simulation times.

Note that the percentages do not add up to 100; the rest of the simulation time is used for other simulation management tasks, such as buffering and data display. The reduction in the sampling frequency also reduces this component of simulation time.

	Traditional method		Efficient method	
Component	Time (s)	Percentage	Time(s)	Percentage
Total	457	100	129.7	100
Simulation time-steps (pulse generation)	94.89	20.8	14.84	10.8
Spreading	9.97	2.2		
Combined spreading, channel modeling and chip-matched filtering			16.45	12.7
Channel estimation	0.55	0.1	0.44	0.3
Multistage detector	301.62	65.9	88.88	68.5

Table 4.2 : **Comparison of profiling data for the two methods of simulation for the system with a multistage detector:** the system with a different detector shows similar behavior as in the previous example. However, the multistage detector takes substantially less time in the new method because an iterative method is employed in modeling the block. These iterations slow down the system with the finer simulation granularity.

Note that the individual time components do not add up to the total simulation time (and hence the percentages do not add up to 100); the rest of the simulation time is used for other simulation management tasks. The reduction in the sampling frequency also reduces this component of simulation time.

Chapter 5

Rapid Prototyping and Simulation Acceleration Using DSPs

This chapter deals with the development of prototyping support for Simulink block diagrams on Digital Signal Processing (DSP) hardware. The motivation for developing DSP support is twofold. Because DSPs are widely used for computations in signal processing and communication systems [23], It is useful to have a method of quick DSP prototyping of algorithms developed using the testbed. Further, the speed of simulation can be improved by complementing the processing power of the host (on which Simulink is running) with additional DSP hardware.

5.1 Prototyping with Real-Time Workshop

Simulink has a component called Real-Time Workshop (RTW), which generates ANSI C-code [24]. The C-code can be compiled using code generation tools for either the host platform or a DSP board. The executable derived from this process can run independently of the Matlab environment. The stand-alone simulations are faster than the Simulink block diagrams using the Matlab engine. The details in developing RTW support for a particular board are given in the appendix(section C.1).

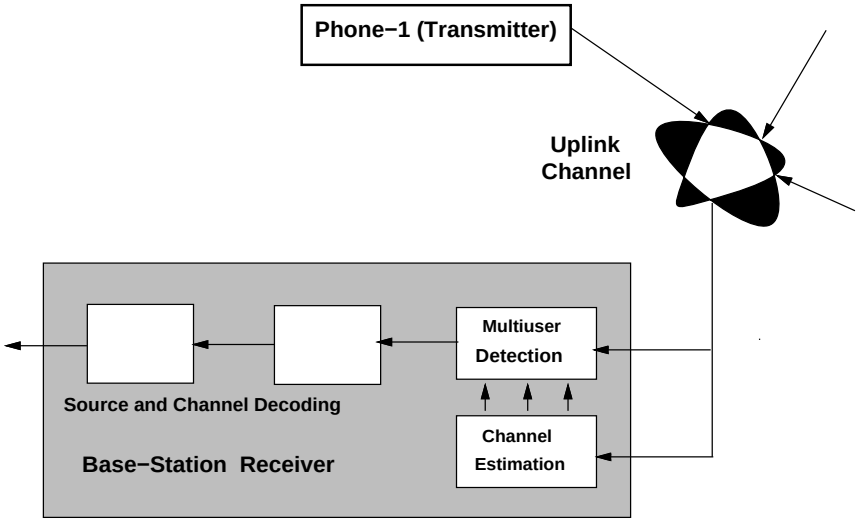


Figure 5.1 : A base-station receiver

5.2 DSP rapid prototyping

RTW is a tool for rapid-prototyping, i.e. quick translation of algorithmic ideas to DSP implementations. The Simulink block diagrams can be used to package some of the algorithms into commercially viable products. For example, the shaded part of Figure 5.1 is essentially the baseband section of a wireless base-station. Therefore RTW can be used to generate DSP code for this part of the system and prototype it on a DSP.

5.3 Simulation acceleration using stand-alone simulators

The simulation speed in environments such as Simulink is slow. The data processing involved is huge because there are many users, and relatively long bit streams are needed for Monte-Carlo simulations. Further, the bits are spread to chips (usually it is 31 or 63 chips per bit in Direct Sequence CDMA systems), and multiple samples are needed for each chip to effectively model the system. The speed of simulation can

be improved by complementing the processing power of the host (The Pentium-2 PC on which Simulink is running) with additional hardware.

The details of RTW support for a DSP board are given in the appendix (section C.1). The process involves creating a *template make file*, and changing references to the cross-compiler (code generation tools). This will enable code to be generated for a Simulink block diagram. The code can then be downloaded on to the DSP board and executed. The returned data is analyzed in the Simulink environment on the host workstation.

5.4 RTW based simulation system for the Pentium host

A basic part of the CDMA physical layer link was modeled (Figure 5.2) in Simulink using components from the Communication and DSP libraries. Some parts of the system such as the spreading block were developed from C-MEX code. The RTW tool was used to generate the C-code for a stand-alone simulator. This code was compiled and linked using Visual C++ tools to create a executable for the Pentium class host.

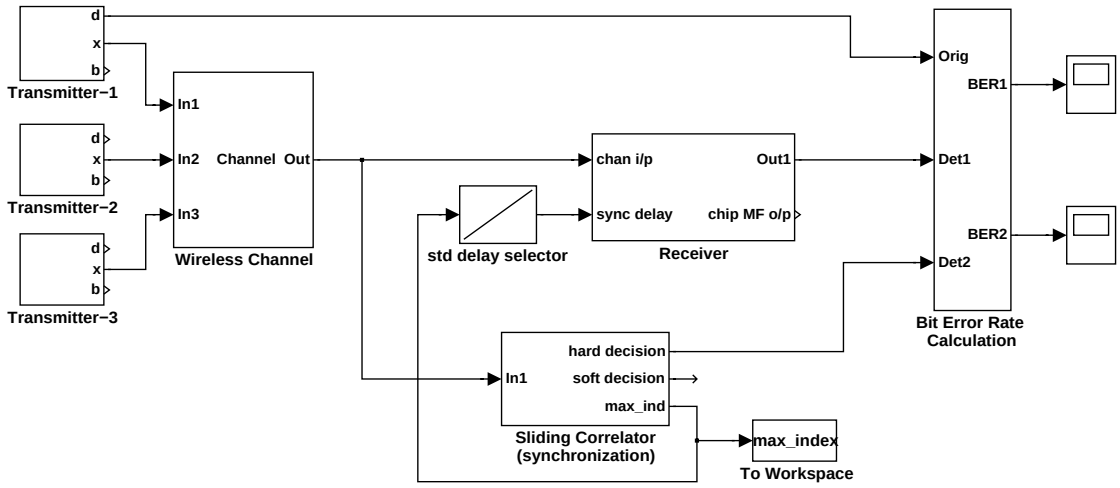


Figure 5.2 : The RTW based simulation (stand-alone simulator on Pentium)

Chapter 6

Future Work

The work in this thesis develops the backbone for a software testbed. The modular features of the system allow its expansion to incorporate more realistic models and complicated algorithms. With support for Real Time Workshop, the Simulink environment will allow the use of DSP hardware to complement the host processor.

6.1 Enhancing the library of blocks

The CDMA library needs to be extended to include more models for channels, receiver structures and coding layers. A realistic channel model needs to include fading (Rayleigh fading/Jakes model). The receiver structure can be extended to have a bigger set of algorithms for synchronization and detection. The maximum likelihood algorithm being used for channel estimation needs to be enhanced for fading channels (with complex data - I and Q channels). A rake receiver is required for multipath channels. The subspace method of channel estimation [25–27] is a proposed inclusion into the library.

The simulation system is intended for evaluation of proposed third-generation Wideband CDMA systems. The system parameters defined by these standards need to be incorporated. The multiuser link presently supports the same data rates for all the users. A useful enhancement will be to provide variable data rates. The work in this thesis has concentrated on the physical layer. The coding layer needs to be

modeled with algorithms for source coding (compression) and channel coding(error control).

6.2 DSP based simulation acceleration

Developing Real-Time Workshop (RTW) support for newer generations of DSP boards will be useful in using them for simulation acceleration (section 5.3). The data intensive computations in the Simulink can be downloaded to a floating point DSP chip. Simulink needs to communicate with the DSP board, get the data back to the Pentium-2 host, interpret and display the results on the PC. Real Time Workshop (RTW) can be used to generate C-Code for the model in Simulink. This code is compiled by the code generation tools specific to the target DSP board. The Texas Instruments' (TI) TMS320C6x digital signal processors [28] will be used in these studies. The code generation tools which include optimizing compilers, assemblers and linkers are provided by TI.

6.3 Multi-Processing

To acheive further acceleration, there is a need for more computation resources, which can be fulfilled by some form of multiprocessing. The options are to use additional DSP boards or have multiple DSPs on a board. The TI software tool called **code composer** is capable of controlling multiple processors connected to the host.

The methods of task decomposition also need to be explored. A possible method could be to partition the entire system (in the Simulink block diagram) into smaller sub-groups and to RTW-compile them separately. The availability of proper tools should enable the execution of these partitions on different boards. A clean mechanism of communication between the different processors needs to be setup.

Another challenge in this work will be to develop a method of multiprocessing for the TI TMSC6x (or C6x for short) generation of processors. The C67x is based on instruction level parallelism. Unlike a previous generation of DSPs – the TMSC4x(which had a communication mechanism between the processors), the C6x generation does not have architectural support for coarse-grained parallelism. Hence multiprocessing support will need to work around these features.

Appendix A

A.1 Listing of system parameters and signals

Parameter	Notation
Number of users	K
Spreading gain	N
Total number of bits in the stream	L_{total}
Total number of bits in the preamble	$L_{preamble}$
Number of paths	P
Signal to noise ratio	SNR
Signal to interference ratio	$SINR$

Table A.1 : Listing of system parameters and corresponding notations.

A.2 Analysis for channel output and the subsequent joint synchronization and detection

This section presents the system model and the analysis for the joint channel estimation and detection problem developed in Sengupta's thesis [13, 14, 17, 18].

Signal/Structure	Notation
Bit sequence	$\mathbf{b}_i$ and $\mathbf{d}$
AWG Noise vector	ν_i
Spreading matrix	$\mathbf{A}$ and $\mathbf{S}$
Amplitude matrix	$\mathbf{W}$
Chip matched filter output vector	$\mathbf{r}_i$
Code matched filter vector	$\mathbf{y}_i$
Estimated spreading matrix	$\mathcal{Y} \stackrel{\triangle}{=} \mathcal{U}\mathbf{Z}$
Cross-correlation matrix	$\mathbf{R}$
Matched filter detector output	$\widehat{\mathbf{d}}$
Decorrelator output	$\widehat{\mathbf{d}}_{decorr}$
Multistage detector (l^{th} stage) soft-decision output vector	$\widehat{\mathbf{z}}^{(l)}_{multistage}$

Table A.2 : Listing of system signals and corresponding notations.

A.2.1 Analysis of the data generation, effect of the channel and the chip matched filtering

Consider a Direct Sequence CDMA system with K users and a spreading code with N chips per bit. BPSK modulation is used with each transmitted signal limited to $[0, T]$. The k^{th} user's transmitted signal is

$$s_k(t) = \sqrt{2P_k} \sum_{i=0}^{L-1} b_{k,i} \times c_k(t - iT) \tag{A.1}$$

where, P_k = transmitted power of k^{th} user; $b_{k,i} \in \{+1, -1\}$ is the i^{th} transmitted bit

in a stream of L bits.

$$c_k(t) = \sum_{n=0}^{N-1} c_{k,n}\Pi(t - nT_c) \tag{A.2}$$

is the spreading waveform, with $c_{k,n} \in \{+1,\ -1\}$ and $\Pi(t)$ being a rectangular pulse of duration T_c; $T = NT_c$, as there are N chips per bit.

The received signal due to the superposition of the attenuated and delayed signals of the K users is

$$r(t) = \sum_{k=1}^{K} w_k \times s(t - \tau_k) + \nu(t) \tag{A.3}$$

where w_k is the complex amplitude with which the k^{th} signal is received and includes the effect of channel attenuation and the phase offset; τ_k is the relative delay with respect to a reference at the receiver. The noise component $\nu(t)$ is assumed to be Gaussian with zero-mean and double-sided spectral density of $\mathcal{N}_0/2$.

This received signal is fed into a chip matched filter whose output is

$$r[n] = \int_{nT_c}^{(n+1)T_c} r(t)dt. \tag{A.4}$$

A discrete observation vector $\mathbf{r}_i$ is formed at bit i by sampling the integrator output and collecting N successive chip matched filter outputs:

$$\mathbf{r}_i = [\ \ r[i]\ \ r[i+1]\ \ \cdots\ \ r[i+N-1]\ \ ]^T. \tag{A.5}$$

Each observation vector $\mathbf{r}_i$ can be viewed as a linear combination of $2K$ signal vectors, corresponding to 2 from each of the K users due to the past and the current bits as in Figure 4.6. So $\mathbf{r}_i$ can be written as

$$\mathbf{r}_i = \mathbf{AWb}_i + \nu_i \tag{A.6}$$

with $\nu_i \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$.

where, $\mathbf{A}$ is the $N \times 2K$ matrix of signal vectors, which depends on spreading codes and delays of each of the users; $\mathbf{W}$ is a $2K \times 2K$ diagonal matrix of complex amplitudes; $\mathbf{b}_i$ is the $2K \times 1$ vector of the K users' previous and current data bits; and the $N \times 1$ vector $\mathbf{K}$, is the noise covariance.

The matrices $\mathbf{A}$, $\mathbf{W}$ and $\mathbf{b}_i$ are expanded below for clarity. The $N \times 2K$ matrix $\mathbf{A}(\tau)$ has columns corresponding to two adjacent bits of each user.

$$\mathbf{A}(\tau) \stackrel{\triangle}{=} [\mathbf{a}_1^R(\tau_1) \;\; \mathbf{a}_1^L(\tau_1) \;\; \cdots \;\; \mathbf{a}_K^R(\tau_K) \;\; \mathbf{a}_K^L(\tau_K)]. \tag{A.7}$$

Let $\tau_k/T_c = q_k + \gamma_k$, $q_k \in \{0, 1, \cdots, N-1\}$, $\gamma_k \in [0, 1]$, so that

$$\mathbf{a}_k^R(\tau_k) = (1-\gamma_k)\mathbf{c}_k^R[q_k] + \gamma_k \mathbf{c}_k^R[q_k+1] \tag{A.8}$$

$$\mathbf{a}_k^L(\tau_k) = (1-\gamma_k)\mathbf{c}_k^L[q_k] + \gamma_k \mathbf{c}_k^L[q_k+1] \tag{A.9}$$

where $\mathbf{c}_k^R[q_k]$ and $\mathbf{c}_k^L[q_k]$ are the spreading codes shifted by integer (multiples of chips) delays.

$$\mathbf{c}_k^R[q_k] = [c_{k,N-q_k} \cdots c_{k,N-1} \; 0 \cdots 0]^T \tag{A.10}$$

$$\mathbf{c}_K^L[q_k] = [0 \cdots 0 \; c_{k,0} \cdots c_{K,N-q_k-1}]^T. \tag{A.11}$$

Expanding $\mathbf{A}(\tau)$ using the above equations gives

$$\begin{aligned}\mathbf{A}(\tau) \;=\; & [(1-\gamma_1)\mathbf{c}_1^R[q_1] + \gamma_1\mathbf{c}_k^R[q_1+1] \;\; (1-\gamma_1)\mathbf{c}_1^L[q_1] + \gamma_1\mathbf{c}_1^L[q_1+1] \;\; \cdots \\ & \cdots \;\; (1-\gamma_K)\mathbf{c}_K^R[q_K] + \gamma_K\mathbf{c}_K^R[q_K+1] \;\; (1-\gamma_K)\mathbf{c}_K^L[q_K] + \gamma_K\mathbf{c}_K^L[q_K+1]].\end{aligned} \tag{A.12}$$

$$\mathbf{A}(\tau) = \begin{bmatrix} (1-\gamma_1)c_{1,N-q_1} + \gamma_1 c_{1,N-q_1-1} & 0 & \dots \\ \vdots & \vdots & \dots \\ (1-\gamma_1)c_{1,N-1} + \gamma_1 c_{1,N-2} & 0 & \dots \\ 0 + \gamma_1 c_{1,N-1} & (1-\gamma_1)c_{1,0} + 0 & \dots \\ 0 & (1-\gamma_1)c_{1,1} + \gamma_1 c_{1,1} & \dots \\ \vdots & \vdots & \dots \\ \vdots & \vdots & \dots \\ \vdots & \vdots & \dots \\ \vdots & \vdots & \dots \\ 0 & (1-\gamma_1)c_{1,N-q_1-1} + \gamma_1 c_{1,N-q_1-2} & \dots \end{bmatrix}$$

$$\begin{bmatrix} \dots & (1-\gamma_K)c_{K,N-q_K} + \gamma_K c_{K,N-q_K-1} & 0 \\ \dots & \vdots & \vdots \\ \dots & \vdots & \vdots \\ \dots & \vdots & \vdots \\ \dots & \vdots & \vdots \\ \dots & (1-\gamma_K)c_{K,N-1} + \gamma_K c_{K,N-2} & 0 \\ \dots & 0 + \gamma_K c_{K,N-1} & (1-\gamma_K)c_{K,0} + 0 \\ \dots & 0 & (1-\gamma_K)c_{K,0} + \gamma_K c_{K,1} \\ \dots & \vdots & \vdots \\ \dots & 0 & (1-\gamma_K)c_{K,N-q_K-1} + \gamma_K c_{K,N-q_K-2} \end{bmatrix}. \quad \text{(A.13)}$$

Similarly,

$$\mathbf{W} = diag[w_1, w_1, \cdots, w_k, w_k, \cdots, w_K, w_K] = \begin{bmatrix} w_1 & 0 & \dots & \dots & 0 & 0 \\ 0 & w_1 & \dots & \dots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \dots & \dots & w_K & 0 \\ 0 & 0 & \dots & \dots & 0 & w_K \end{bmatrix} \tag{A.14}$$

and,

$$\mathbf{b}_i = [b_{1,i-1}, b_{1,i}, \dots, b_{K,i-1}, b_{K,i}]^T \tag{A.15}$$

where $b_{k,i}$ is the i^{th} bit of the the k^{th} user.

A.2.2 Channel parameter estimation and detection

The equation for the received chip matched filter output A.6 is re-written as in the equation A.16. This method accommodates the effects of multiple paths without increasing the size of the matrices $\mathcal{U}$ and $\mathbf{Z}$, but making them more dense.

$$\mathbf{r}_i = \mathcal{U}\mathbf{Z}\mathbf{b}_i + \nu_i \tag{A.16}$$

where $\mathcal{U} = [\mathcal{U}_1^R \ \mathcal{U}_1^L \cdots \mathcal{U}_k^R \ \mathcal{U}_k^L \cdots \mathcal{U}_K^R \ \mathcal{U}_K^L]$ and $\mathbf{Z} = diag(\mathbf{z}_1, \mathbf{z}_1, \cdots, \mathbf{z}_K, \mathbf{z}_K)$; with $\mathcal{U}_k^R = I_M \otimes \mathbf{U}_k^R$ and $\mathcal{U}_k^L = I_M \otimes \mathbf{U}_k^L$; the operator '$\otimes$' represents the Kronecker product of two matrices and I_M is the M dimension identity matrix; Further $\mathbf{U}_k^{(R)} =$

$[\mathbf{c}_k^{(R)}[0] \cdots \mathbf{c}_k^{(R)}[N-1]$ and $\mathbf{U}_k^{(L)} = [\mathbf{c}_k^{(L)}[0] \cdots \mathbf{c}_k^{(L)}[N-1]]$ are the matrices formed from the spreading codes delayed by all possible integer delays. For the single sensor case considered here,

$$\mathbf{z}_k = \begin{bmatrix} 0 \\ \vdots \\ w_{k,1}(1-\gamma_{k,1}) \\ w_{k,1}\gamma_{k,1} \\ \vdots \\ w_{k,P}(1-\gamma_{k,P}) \\ w_{k,P}\gamma_{k,P} \\ \vdots \\ 0 \end{bmatrix}. \tag{A.17}$$

It is shown in Sengupta's thesis [13, 14, 17, 18] that the channel parameters extracted using Maximum Likelihood estimation can be captured in a single matrix $\mathcal{Y} \triangleq \mathcal{U}\mathbf{Z}$. This matrix is estimated as $\widehat{\mathcal{Y}}$, where

$$\widehat{\mathcal{Y}} = \widehat{\mathbf{R}}_{rb}\widehat{\mathbf{R}}_{bb}^{-1}, \tag{A.18}$$

and

$$\widehat{K}(\widehat{\mathcal{Y}}) = \widehat{\mathbf{R}}_{rr} - \widehat{\mathcal{Y}}\widehat{\mathbf{R}}_{rb}^H \tag{A.19}$$

with $\widehat{\mathbf{R}}_{rr} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{r}_i\mathbf{r}_i^H$, $\widehat{\mathbf{R}}_{br} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{b}_i\mathbf{r}_i^H$, and $\widehat{\mathbf{R}}_{bb} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{b}_i\mathbf{b}_i'$. Further, the estimate of this matrix $\mathcal{Y}$ can be used to find the matched filter detector output vector $\mathbf{y}_i$ using

$$\mathbf{y}_i = (\mathcal{U}\mathbf{Z})^T \mathbf{r}_i \tag{A.20}$$

where, $\mathbf{r}_i$ is the chip matched filter output vector. Further, the $\mathcal{U}\mathbf{Z}$ matrix is can be used to generate the cross correlation matrix using

$$\mathbf{R} = (\mathcal{U}\mathbf{Z})^T (\mathcal{U}\mathbf{Z}). \tag{A.21}$$

Appendix B

B.1 Construction details of the Simulink based simulation system

This section presents the details of the construction of the Simulink based system [12]. The major blocks described are data generation, channel models, channel estimation, and multiuser detection. The components were derived from the Simulink library, and the custom built library developed from C-MEX or Matlab file based S-functions [22]. There is a global initialization file (`glob_gen.m`) which is called during updating of parameters. This file creates a Graphical User Interface (GUI) for setting the workspace variables corresponding to parameters such as number of users (K), spreading gain (N), number of bits in the stream ($L_{preamble}$ and L_{total}) and number of paths (P).

B.1.1 Automated netlist generation

This method uses Matlab functions such as `add_block, delete_block, replace_block, add_line, delete_line, set_param` to reconfigure the Simulink block diagram according to the system parameters. For example, the data generation block is created by automatic netlist generation. If a user is added to the system, the data blocks necessary for this user are added and are connected to the rest of the system. If a different algorithm is chosen for a block (eg. multiuser detection), the new block replaces the old one.

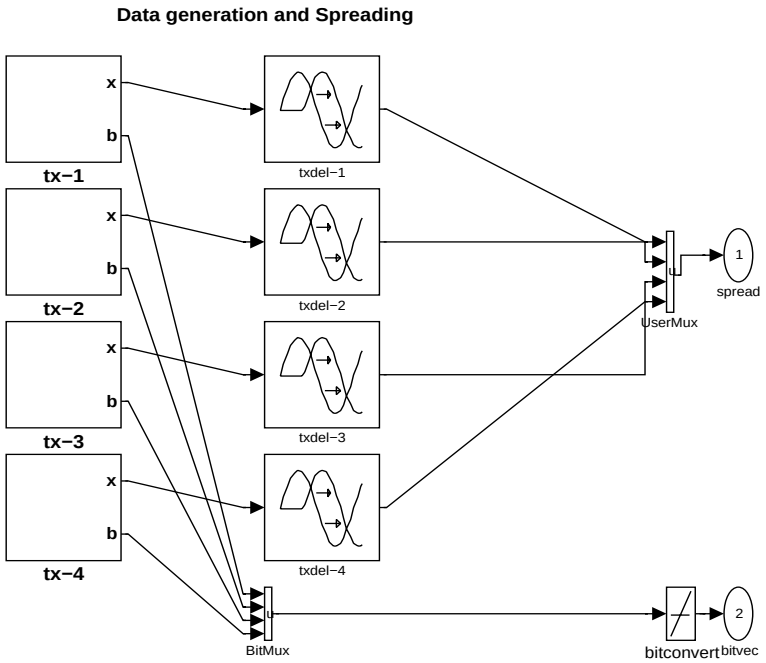


Figure B.1 : Construction of the data generation block with general spreading and arbitrary delays

B.1.2 Data generation, spreading and channel modeling

Figure B.1 shows the sub-system with different bit streams being spread according to the spreading matrix. This data generation block is created by automatic netlist generation. The spreading is done by an S-function(`sprd.m`) by multiplying the bit(1 or -1) with the spreading sequence for that user. The spreading matrix is initialized as a variable in the workspace at the time of an update, according to the value of K and the type of spreading (Gold-codes, M-sequences or random spread). The spreading matrix is created by a table lookup for Gold codes and M-sequences. It is created by a random matrix generation function for the random spreading codes. The same spreading matrix is used in the channel estimation block also.

The channel block, blown up in Figure B.2, uses a delay block (Figure B.3) which

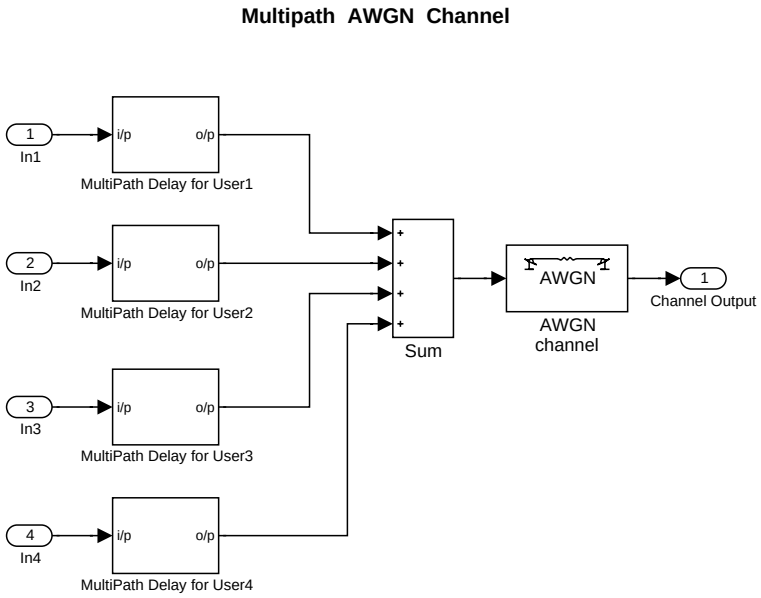


Figure B.2 : The multipath AWGN channel

delays each users' data with the different path delays. The signals of different users are superimposed and AWG noise is added according to the Signal-to-Noise ratio (SNR) value in the parameter.

The chip-matched filtering shown in Figure B.4 is required at the front end of the receiver. The discretized output of the channel is integrated over the chip duration and dumped at the end of it. The integrator output is held before the dump and presented as the chip matched filter output.

B.1.3 Maximum Likelihood synchronization

Figure B.5 shows the simulink implementation of the Maximum Likelihood channel estimation. A buffer collects the $N \times L$ chip matched filter outputs and the L preamble bits of the K users. These are processed in the S-function `S_ml_chan_est.m` to estimate the $\mathcal{U}\mathbf{Z}$ matrix, which is a compact representation of the channel parameters.

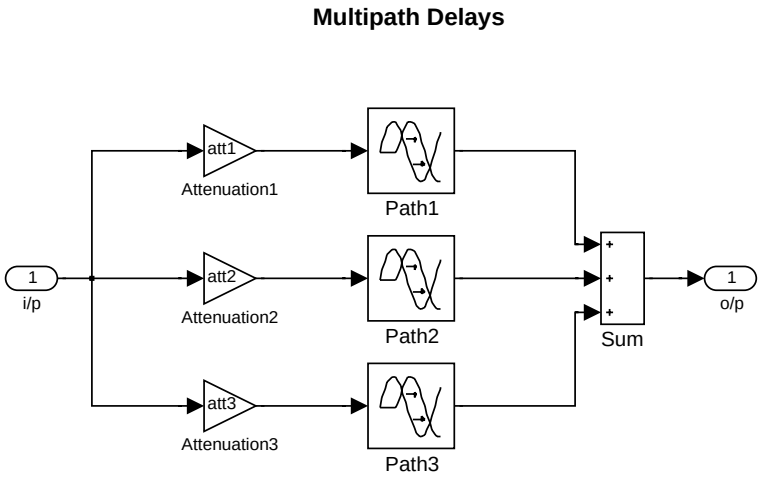


Figure B.3 : The delay block for the multipath channel

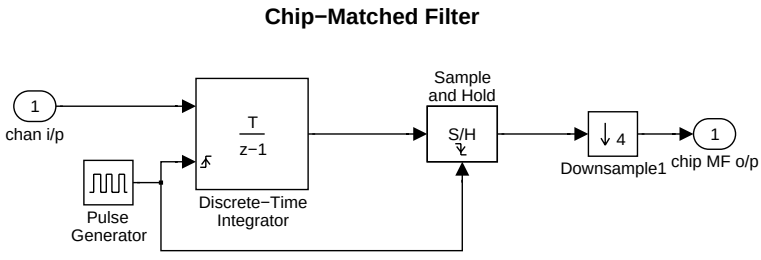


Figure B.4 : The chip matched filter

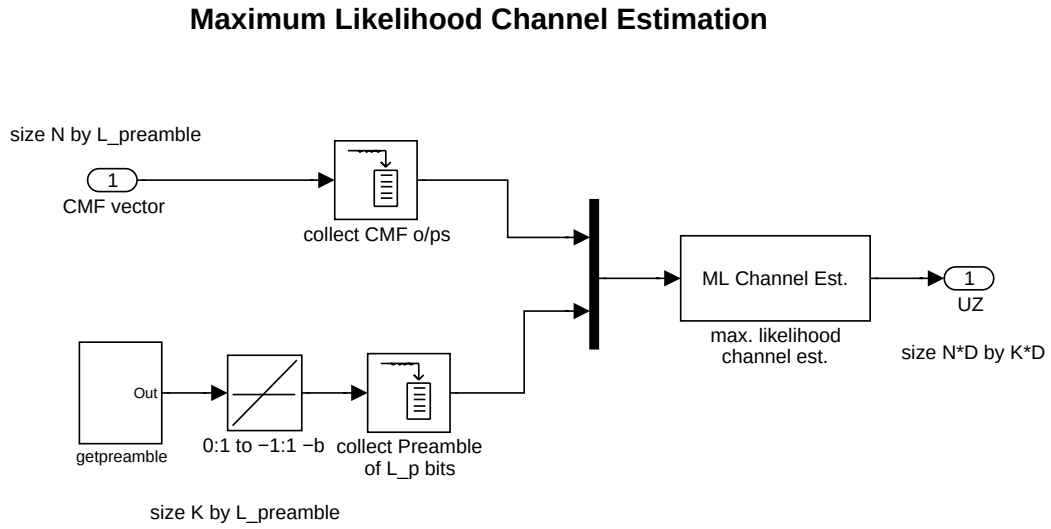


Figure B.5 : The channel estimation block

B.1.4 Detector blocks

The construction of the Multiuser detector block is shown in Figure B.6. The chip matched filter output is buffered into a ND vector Y_{cmf} where N is the spreading gain and D is the detection window size. An S-function is used to calculate the cross correlation matrix $R = (\mathcal{U}\mathbf{Z})^T * \mathcal{U}\mathbf{Z}$ of size KD by KD. The detector block is a triggered subsystem shown in Figure B.7. This subsystem calculates the KD vector $Y = (\mathcal{U}\mathbf{Z})^T * Y_{cmf}$ which corresponds to the code matched filter output. The vector Y is then used in the S-function for multiuser detection (`sMatchedFilter.m`, `sDecorr.m and sDiffstage.m respectively`) to reduce the multiple access interference. The detected output is unbuffered and presented at the output as a bit stream.

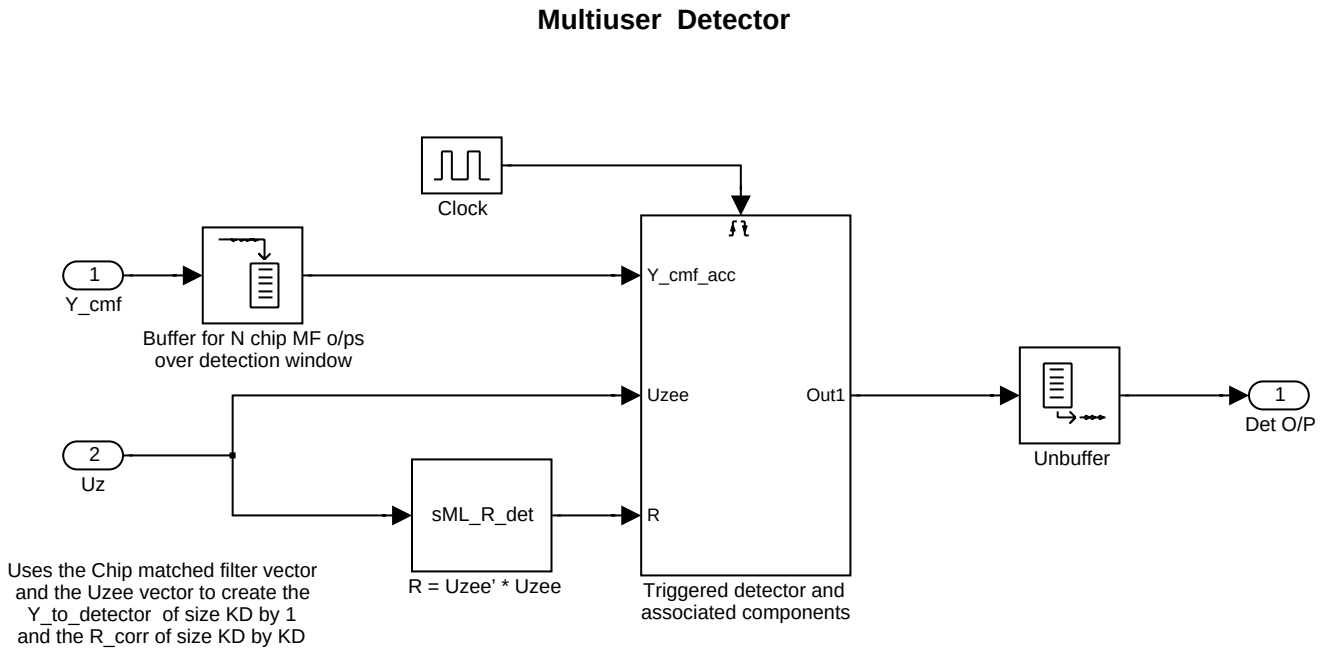


Figure B.6 : The multiuser detector block

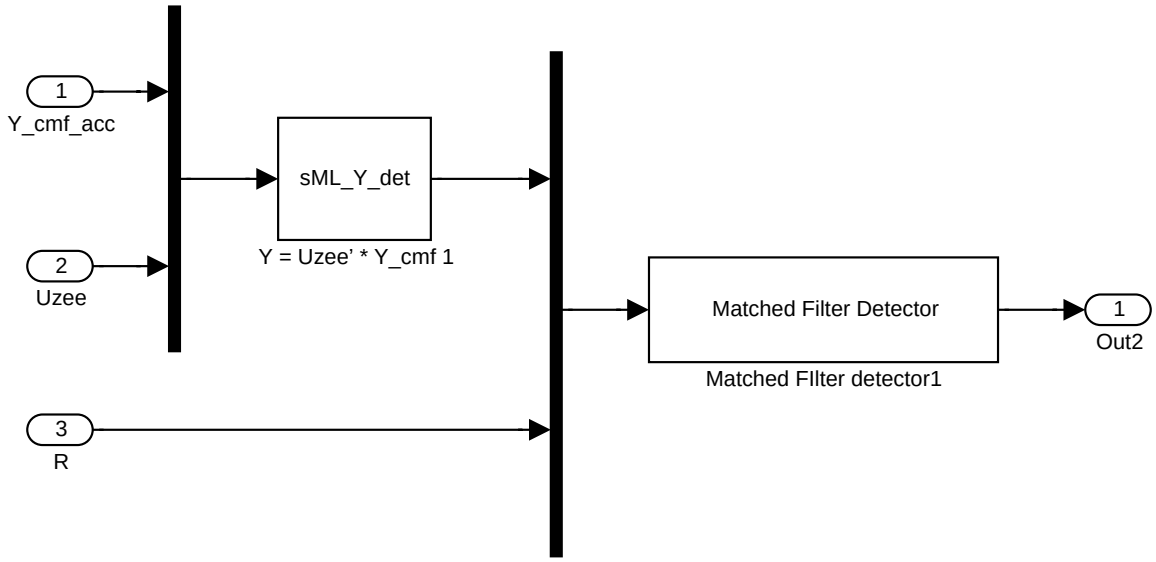


Figure B.7 : The triggered part of the multiuser detector

B.2 Profiling

The Matlab and Simulink environment has profiling support using the `profile` command. When the profiling is turned on, the calls to the various functional blocks are timed. The underlying S-functions are also individually timed in the sequence of block execution. The profiling process generates a report at the end of the run. It gives a complete break-up of the execution time spent in different parts of the simulation system. It also gives a listing of the core part of the code which takes a substantial fraction of the simulation time.

Appendix C

C.1 Board support for C-Code generation with Real Time Workshop

This section gives some details about the method of developing board support for a new DSP board [24, 29]. Consider the simple model (`mult.mdl`) shown in Figure C.1, with two constant matrices being multiplied using the matrix multiplication S-function from the DSP blockset(Figure C.2, and write it into a `mult.mat` file. The RTW build (for the PC) results in a `mult.rtw`, from which the `mult.c`, `mult.h`, `mult.prm`, `mult.reg` and `mult_export.h` are created. This code from the Target Language Compiler (TLC) depends on the `grt.tlc` file. The make file (`mult.mk`) is generated from the `grt_vc.tmf` after substituting some of the file-system paths. This make file is used by NMAKE for building the executable for the PC.

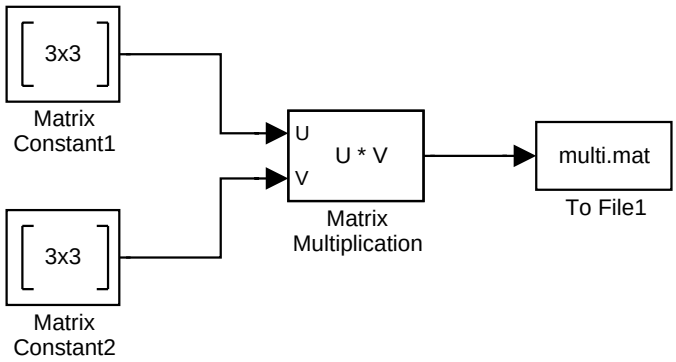


Figure C.1 : The simulink block diagram for matrix multiplication

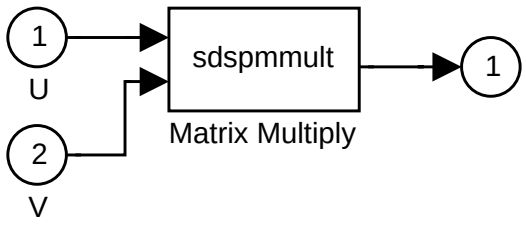


Figure C.2 : The underlying C-MEX S-function for matrix multiplication

This process can be extended for use with another DSP board. The Texas Instruments (TI) C67 EVM is the target board in the following description. The main changes are in the template make-file (`MATLABROOT/rtw/c/grt/grt.tmf`), which is saved in a new directory `MATLABROOT/rtw/c/ti-grt/` as `ti-grt.tmf`. A duplicate of `grt.tlc` is made as `ti-grt.tlc`. The `tlc` file is used by the target language compiler for code generation. The file `vctools.mak` in the directory `MATLABROOT/rtw/c/tools/` is copied into `ti-tools.mak`. The compiler assignments are changed from those of Visual C++ to TI tools. While running a simulink-RTW model on a DSP board instead of a PC, the RTW options should be set to use `ti-grt.tlc` and `ti-grt.tmf` instead of the defaults.

Bibliography

[1] T S Rappaport, B D Woerner, and J H Reed, *Wireless Personal Communications*, Kluwer Academic Publishers, 1996.

[2] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein, "Theory of Spread Spectrum Communications – A Tutorial," *IEEE trans. Communications*, vol. COM-30, no. 5, pp. 855–884, May 1982.

[3] Joseph Cavallaro and Behnaam Aazhang, "Development of a Testbed for Wireless Multiuser Comunication Systems," Project Definition Document (See CMC Web Page), 1998.

[4] WINLAB at Rutgers Univ., "Wireless Propagation and Protocol Evaluation Testbed (WiPPET)," Website – http:// www.winlab.rutgers.edu/ pub/ projects/ Fall98/ Modeling_and_Simulation.html, 1998.

[5] Hewlett Packard Labs, "Simulated BER Performance of, and Initial Hardware Results from, the Uplink in the U.K. LINK-CDMA Testbed," Tech. Rep., http://www.hpl.hp.com/techreports/96/HPL-96-80.html, May 1996.

[6] Center for Multimedia Communications at Rice University (C M C), "http://www.ece.rice.edu/COMM," C M C Web Page, 1999.

[7] Shimon Moshavi, "Multiuser Detection for DS-CDMA Communications," *IEEE Communications magazine*, pp. 124–136, Oct. 1996.

[8] Kurt Matis, "Design, Simulation, Analysis of CDMA Communication Systems," Tech. Rep., ICUCOM corporation, 1997.

[9] Cadence Inc. Alta Group, "Signal Processing Worksystem (SPW) at http://www.cadence.com/alta/products/," Cadence Website, 1997.

[10] Synopsys Inc., "COSSAP at http://www.synopsys.com/products/dsp/dsp.html," Website, 1997.

[11] Mathworks Inc., "Simulink at http:/www.mathworks.com/products/simulink/," Website, 1997.

[12] Simulink group, "Using Simulink," Tech. Rep., The Mathworks, 1999.

[13] Chaitali Sengupta, *Algorithms and architectures for channel estimation in wireless CDMA communciation systems*, Ph.D. thesis, Rice University, Dec. 1998.

[14] Chaitali Sengupta, Joseph Cavallaro, and Behnaam Aazhang, " On Multipath Channel Estimation for CDMA systems Using Multiple Sensors," Submitted to IEEE Trans. of Communications.

[15] R. Gold, "Optimal Binary Sequences for Spread Spectrum Multiplexing," *IEEE proc. Information Theory*, Oct. 1967.

[16] J. G. Proakis, *Digital Communications*, McGraw-Hill, 1995.

[17] Chaitali Sengupta, Ari Hottinen, Joseph Cavallaro, and Behnaam Aazhang, "Maximum Likelihood Multipath Channel Parameter Estimation in CDMA systems," in *32nd Annual Conference on Information Sciences and Systems(CISS)*, 1998.

[18] Chaitali Sengupta, Joseph Cavallaro, and Behnaam Aazhang, ," in *9th IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications(PIMRC)*, 1998.

[19] S. E. Bensley and B. Aazhang, "Maximum likelihood synchronization of a single user for code division multiple access communication systems," *IEEE Trans. on Communications*, vol. COM-46, no. 3, pp. 392–399, Mar. 1998.

[20] Mahesh Varanasi and Behnaam Aazhang, "Multistage Detection in Asynchronous Code-Division Multiple-Access communications," *IEEE trans. of Communications*, vol. 38, no. 4, pp. 509–519, Apr. 1990.

[21] Gang Xu and Joseph Cavallaro, "Real-time implementation of multistage algorithm for next generation wideband CDMA systems," in *Advanced Signal Processing Algorithms, Architectures, and Implementations IX, SPIE*, 1999.

[22] Simulink group, "Writing S-functions," Tech. Rep., The Mathworks, 1999.

[23] Z. Kostic and S. Seetharaman, "Digital Signal Processors in Cellular Radio Communications," *IEEE Communications Magazine*, pp. 22–35, Dec. 1997.

[24] Simulink Group, "Real-Time Workshop Users Guide," Tech. Rep., Mathworks Inc., 1999.

[25] S. E. Bensley and B. Aazhang, "Subspace-Based Channel Estimation for Code Division Multiple Access Communication Systems," *IEEE Trans. on Communications*, vol. 44, no. 8, pp. 1009–1020, Aug. 1996.

[26] Kishore Kota, *Parallel algorithms and architectures for near-far resistant CDMA acquisition*, Ph.D. thesis, Rice University, May 1996.

[27] Chaitali Sengupta, Kishore Kota, and Joseph Cavallaro, "Parallel algorithms and architectures for subspace based channel estimation for CDMA communication systems," in *Advanced Signal Processing Algorithms, Architectures, and Implementation VI*, 1996.

[28] Texas Instruments (DSP Products), "The TMS320C6x generation of DSPs at http://www.ti.com/sc/docs/products/dsp/c6000/," Website, 1998.

[29] Simulink Group, "Target Language Compiler Reference Guide," Tech. Rep., Mathworks Inc., 1999.