The Implementation of a CDMA System on an FPGA-based Software Radio

By Timothy Ellis

University of Natal

2000

Submitted in fulfilment of the academic requirements for the degree of Master of Science in Engineering in the School of Electrical & Electronic Engineering, University of Natal, Durban

December 2000

Preface

Timothy Ellis performed the research work presented in this thesis under the supervision of Professor Roger Peplow, at the University of Natal's School of Electrical and Electronic Engineering in the Centre of Excellence in Radio Access Technologies.

Parts of this thesis have been presented by the student at the SATNAC 1999 conference in Durban, South Africa, and the COMSIG 2000 conference in Somerset West, South Africa.

The entire thesis, unless specifically indicated to the contrary in the text, is the student's own work and has not been submitted in whole or in part to any other university. A full listing of the VHDL, C and C++ source code used in this thesis is available on request.

Acknowledgements

This work was performed for the Radio Access Technologies Centre of Excellence at the School of Electrical and Electronic Engineering, University of Natal, Durban. This centre receives its sponsorship from Telkom, Alcatel Altech Telecomms and THRIP. I would like to thank these members for their support of this research. I would like to extend this thanks to those responsible for the provision of the Flexible Radio Platform at Alcatel. Without this software radio, this project could not have become a reality.

I would like to extend a special thanks to my supervisor, Professor Roger Peplow, for his direction during the project.

Synopsis

This dissertation examines two of the rising technologies in the world of wireless, cellular communications – CDMA and the software radio. This thesis covers many of the issues related to these two emerging fields of wireless communications, establishing a theoretical framework for the broader issues of implementation. To this end, the thesis covers many of the basic issues of spread spectrum communications, in addition to establishing the need for, and defining the role of, the software radio. Amalgamation of these two key areas of interest is embellished in a presentation of many of the concerns of implementing a specific CDMA system on a particular type of software radio – the Alcatel Altech Telecomms Flexible Radio Platform.

Of primary concern in the research methodology embraced in this thesis is the mastering of a variety of analysis and implementation tools. Once the theoretical background has been substantiated by current expositions, the thesis launches along a highly deterministic route. First, the research issues are tested in a mathematical environment for suitability to the given task. Second, an analysis of the appropriateness of the technique for the software radio environment is undertaken, culminating in the attempted deployment within the hardware environment. Rigorous testing of the input/output mapping characteristics of the hardware instantiations created in this manner complements the research methodology with a viability study. This procedure is repeated with many elements of the CDMA system design as they are examined, simulated, deployed and tested.

Glossary

ADC	Analogue to Digital Converter
AMPS	Advanced Mobile Phone Service
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
Bit	A binary digit taking on the value 1 or 0
BUS	A collection of signal-carrying wires allowing processors and peripherals to exchange information
Byte	Sequential collection of 8 bits
CDMA	Code Division Multiple Access – wireless spread spectrum communications technique
Chip	Element of a sequence that is used to represent a bit in a DS-CDMA system. Capable of taking on two levels
DAC	Digital to Analogue Converter
DS-CDMA	Direct Sequence CDMA, using an orthogonal, high-rate sequence
DSP	Digital Signal Processor
FIR	Finite Impulse Response – a DSP filter type
FPGA	Field Programmable Gate Array
GSM	Global System for Mobile Communications
HCIL	Host Communication Interface Library
IF	Intermediate Frequency
IIR	Infinite Impulse Response – a DSP filter type
LUT	Look-Up Table
MAC	Multiply and Accumulate – common DSP operation
MAI	Multiple Access Interference
Mbps	Mega bits per second – 1 024 000 bits per second
Mcps	Mega chips per second – 1 024 000 chips per second
MMSE	Minimum Mean Square Error – a technique used in some MUD's
MUD	Multi-User Detector
NRZ	Non-Return to Zero – series of bi-polar pulses oscillating about zero
PCI	Peripheral Component Interconnect
PN Sequence	Pseudo-Noise sequence often used in the chipping process
Processing Gain	The ratio of the chipping rate to the base-band data rate
RF	Radio Frequency
RMS	Root-Mean Square – measure of the level of a signal
RS-485	Differentially paired physical layer for serial communications
SNR	Signal to Noise Ratio

TDMA	Time Division Multiple Access
TTL	Transistor-Transistor Logic
VHDL	Very High Speed Integrated Circuits Hardware Description Language
VHDL Architecture	Process-level description of a VHDL module
VHDL Entity	Port-level description of a VHDL module
W-CDMA	Wideband CDMA

Contents

List of Figures

List of Tables

Chapter 1 Introduction

With a burgeoning plethora of wireless communication standards pervading the world today, and with no end in sight, communications developers will soon have to reach a previously unseen level of flexibility to remain competitive. This thesis is a presentation of two key concepts in the current trends of wireless communications. First, the emerging protocol of CDMA is examined, highlighting the broadening gap between the protocols of yesterday, and those that will form the basis of tomorrow's technology. This raises the pertinent question of how developers can deal with the coexistence of a variety of protocols amongst networks, highlighting the second major issue that this thesis broaches – that of the software radio. Here lies the promise of flexibility that may be able to overcome protocol diversity in the form of self-adapting mobile communication devices.

The key wireless communications elements broached within this thesis are not without their own eventful histories. A fundamental aspect of these histories is the set of diverse factors that have contributed in the evolution of wireless communications towards its present state. From its humble beginnings as a set of experiments devised by Guglielmo Marconi, to its current mass consumer appeal and high technological status, wireless communications has undergone exponential expansion and acceptance as an integral part of our lives. Consumer statistics, with associated projections, reveal astounding growth in this realm. For instance, from 1990 to 2000, there was an escalation in the number of world mobile subscribers from 30 million to 500 million. It is anticipated that this will increase to 1 billion by the year 2006. This projected time span is minimal compared with the 130 years it took fixed telephone subscriber numbers to reach the same level. With no foreseeable factors contributing to the decline of consumer demand for personal communication devices, wireless communications companies will have to find ingenious tools and strategies to keep up in associated research and deployment. However, before these issues are tackled in the body of this thesis, it is important that a context is established. While it is clear that wireless personal communications devices will continue to gain support, it is important that a framework is established, documenting the progression towards this state. This will not only justify the course of this thesis, but also aid the reader in projecting tomorrow's possibilities within today's technology.

1.1 Progression

Cellular communications is the essential driving force of this thesis - a mobile communications technique that has experienced unprecedented growth in the past decade. During this period, there has been much progression towards improving the state of cellular communications. This improvement has benefited the two key players in the field – the providers, seeking easier means of more cost-effective production and expandability, and the consumer, seeking and

discovering service flexibility and value. This thesis will broach two current wireless communications issues that have been spurred on by the cellular revolution – first, CDMA and second, its use in a software radio context. Both of these concepts are fairly new to the consumer wireless world, and both have an associated history highlighting wireless progression towards their incorporation into contemporary wireless consumer networks.

Code Division Multiple Access (CDMA) is a spread spectrum wireless communications protocol that is gaining widespread acceptance and use in commercial networks. It has many attributes that make it ideal for a multiple user environment. However, in 1998, it comprised only 7% of protocols used in wireless networks, whereas GSM (the Global System for Mobile Communications) dominated with a staggering 44%. Further, these second-generation digital cellular services have been in existence since 1989. In this light, CDMA certainly seems to have played little or no role in spurring the personal wireless communications world to its current state, and one may even ask why there should be a departure from the current situation. The answer lies wrapped up in two insatiable demands in the wireless domain – those of data rates and spectrum preservation. A paradox lies in the meeting of these demands in that a higher data rate generally requires a higher spectrum usage. CDMA is one technology that promises to overcome some of the issues related to the limitations of existing technologies. However, the progression towards its incorporation into the current fold of protocols has involved a lengthy pioneering period, incorporating the contributions of many renowned communications experts. In his detailed overview of the subject, [Prasad, 1998], [1], includes a snapshot of the process. The evolution of CDMA began with an initial concept, proposed by John Pierce in 1949. Here, a simple time hopping spread spectrum multiple access system was described, and laid the foundation for future research. Also in 1949, Robert Pierce and Cluade Shannon expounded two core elements of CDMA systems, those of the interference averaging effect and graceful degradation of the system under increased access. Further, Magnuski first mentioned one of the fundamental stumbling blocks of spread spectrum systems, the near-far effect, only in 1961. At this point, the value of spread spectrum was clear, since the mathematical analysis proved the value of the technology. However, CDMA presents many technological challenges, even for today's network designers. As a result, there was little penetration into a consumer context, and the advantages that spread spectrum offered communications systems subject to interfering signals (or jamming signals) were exploited primarily by the military for field and navigation purposes. Even this occurred only in the 1970's. It was only in the late 1970's that the first proposals were brought forward to include spread spectrum technology in civilian cellular communications.

The 1980's saw developments in CDMA technology leading to commercial implementation of a CDMA cellular network. In 1986, Verdu proposed an optimal multi-user detector and, as his results show in [2], [Verdu, 1986], performance can be made to approach that of a single user channel. Subsequent to this groundbreaking advancement, there has been copious research into

the design and optimisation of these detectors, and this thesis also examines some of the associated implementation issues. However, it was not until 1993 that CDMA was standardised as a cellular communications technology. In that year, the IS-95 standard proposed a narrowband form of CDMA, with a 1.25MHz channel bandwidth, which was finally implemented in a commercial context only in 1996. Since this successful deployment, it has been realised that to attain the desired high data throughput of many of the proposed multi-media cellular applications, the bandwidth of CDMA has to be increased. This has led to the development of wideband CDMA (W-CDMA) that has seen several standards emerging in countries such as the United States (cdma2000), Japan (Core-A) and Europe (FRAMES FMA2). In these systems, to achieve data rates approaching 1 Mbps, it is generally accepted that the channel bandwidth will have to be at least 20MHz. However, the commercial deployment of W-CDMA looks set only to occur between 2001 and 2002. In spite of the standards that emerge for the networks of the near future, it is clear that, if they follow current trends, they are certain to have a strong CDMA foundation.

As one follows the history of the developments in wireless communications, it becomes apparent that two fundamental obstacles prohibited more rapid deployment. These come in the form of very tight regulations regarding the allocation of spectrum resources, coupled with economic and business issues. However, as one traces the penetration of protocols into the cellular arena, a very real advantage is uncovered in the form of technological progression. The delay in cellular deployment has seen advances in microprocessors and associated hardware, which means that today's cellular networks have an exceptional technological basis for development. Further, as protocols have begun to penetrate commercial implementations, so the underpinning technology has already been in place. This development has seen several generations of cellular flavours emerging in the world markets.

The first generation of cellular communications began in Tokyo in 1979 with the successful commercial installation of a cellular system, as noted by SRI International in [4]. Soon after this, the Advanced Mobile Phone Service (AMPS), a North American implementation of the analogue IS-19 standard, emerged. Here, the channels were separated by only 30kHz. Since the mobile units were analogue, they suffered from a high degree of inflexibility. A decade after the first appearance of the cellular system, the second generation was ushered in with the introduction of a superior digital counterpart - GSM. GSM has gained much of today's cellular market and is a highly successful protocol. A new standard, IS-95, embodying the nature of narrow band CDMA followed GSM in 1993. Many of the benefits of CDMA, explaining its emergence at the height of GSM, are outlined in this thesis. However, IS-95 is incapable of meeting the escalating demands of cellular subscribers, and the third generation of cellular concepts was born to broach these shortcomings. Extending initial CDMA attempts by increasing the bandwidth, wide-band CDMA will more than likely pervade tomorrow's networks. However, not only are there several generations of protocols already in existence,

but there are several variations as one travels around the world. This protocol variation becomes a severe hindrance to wide-range mobility, especially in the event that the mobile unit's functionality is tied to its hardware. This is generally the case with today's mobile units where they are built to comply with a specific standard and fail in a network making use of a different one. Further, the development and testing of units to handle new standards also becomes difficult for mobile telecommunications companies. Thus many are turning to the promise of the software radio, an item finding its beginnings in a predominantly non-civilian context.

The key to flexibility in the development and construction of mobile cellular units lies in the digital domain. Reproducibility of a digital design is far greater than its analogue counterpart. However, it is still possible for a digital design to be completely inflexible in its capabilities. The software radio is a device that overcomes this limitation by moving its functionality to a re-configurable digital block. This thesis will examine many of the issues related to a software radio. Historically, the software radio has been used since the early 1980's when electronic warfare first emerged. Today's military requires a highly versatile communications device, especially in the case of international collaboration. Military software radios can be configured to adhere to any one of many protocols and seamlessly integrate into a tactical network. The rise of the Digital Signal Processor (DSP) and other high-speed digital logic has spurred the development of these devices and allowed them to penetrate the commercial context. As can be seen from the number of emerging communications protocols discussed so far, the software radio is a vital tool in the provider and subscriber realms. Developing units for new protocols is now far easier, as a reusable platform can be employed for several projects. For the subscriber, it is possible to have one mobile unit that can exist seamlessly in several networks, each with a different communications standard. This thesis will examine a particular type of a software radio. Simultaneously, the implementation of a specific CDMA system on the radio will allow the reader to see how the radio's flexibility can work towards easier development processes.

1.2 Chapter Outline

Chapter 2 introduces the concept of Code Division Multiple Access (CDMA). This chapter will allow the reader to gain an understanding of one of the fundamental building blocks of this project. CDMA will be introduced not only within the context of its importance as a wireless communications protocol, but some of its foundation maths will allow the reader to begin to conceptualise the protocol in terms of implementation issues.

Chapter 3 unveils yet another core issue - multi-user detection - and is a logical extension to the discussion begun in chapter 2. Multi-user detection will be expounded within the framework of Direct Sequence (DS) CDMA. Some of the options available will be presented as a literature survey, while the method implemented in this thesis – that of the minimum mean square error (MMSE) multi-user detector – will be examined in detail.

Chapter 4 familiarises the reader with the software radio platform and begins to pull the reader's attention more towards the practicality of the project while beginning the discussion on some of the more fundamental implementation issues. First, the software radio is defined and conceptualised, while a few variations and extensions are explored. Second, the Alcatel Altech Telecomms flexible radio platform is discussed, giving the reader a firm understanding of the environment in which the thesis was implemented.

Chapter 5 moves closer towards the critical implementation issues, this time examining in detail the relevant design and synthesis issues. The reader will gain a better understanding of VHDL as a means of describing, and thereby designing, complex digital circuitry. The various design methodologies will be expounded and put in context. Primarily, this chapter will seek to develop the relationship between the style of design and the synthesis tool. These synthesis issues naturally determine the boundaries within which the developer must operate. To this end, data manipulation is examined, and placed firmly within the limitations of the synthesis tools. The data sources for the FPGA include Analogue to Digital converters and serial streams. Since this data needs to be stored before it can be processed, and this manipulation needs to occur at high speeds, this chapter also examines suitable data storage techniques and associated data types.

Chapter 6 begins to describe the actual system that has been implemented, giving an overview of the major functional blocks. This chapter will help the reader to place each element in context, as they are unveiled in successive chapters.

Chapter 7 moves the reader into the implementation arena, tackling one of the primary functional blocks proposed in the preceding chapter. The software radio requires a serial stream to port control and data to and from an external source. In the case of the project, this external data source is a personal computer with a DSP-based PCI communications card. Critical to the system is that the interface between the PC and the software radio is well defined, allowing for messages to be passed correctly between the two layers. This chapter gives a detailed description of the interfacing issues broached in the thesis. Without this chapter, the system operates in isolation, and examining it closely reveals and solves some of the associated problems. The chapter familiarises the reader with the links between the DSP subsystem and the software radio. This involves a standard communications bus (the SC-BUS) that is a high-speed serial link. Further, this link is converted to RS-485 to allow for transmission over long lines. As some of the bus timing specifications are detailed, so the reader is made aware of some interfacing complexities. The power of VHDL is also explored, showing its use in overcoming many of these difficult timing issues.

Chapter 8 explores one of the core elements of the transmission aspect of any CDMA system – that of spreading the base-band data. In a direct-sequence CDMA system, each user's data bit is replaced with a higher frequency series of bits, known as a code. This process of spreading is

also commonly termed "chipping". Chapter two embellishes the subject of codes, unveiling variations such as Walsh and Gold codes. This chapter extends that theoretical base by proposing a particular means of implementing a Walsh encoder. This chipping entity is subject to critical timing constraints that are examined in the context of cross-chip timing restrictions. Further, the issues related to superposing the data of several users, and thereby simulating the nature of a base station, is also unveiled.

Chapter 9 will allow the reader to gain an understanding of what became a major portion of the implementation cycle – that of base-band pulse shaping. This normally DSP-based operation has been explored at length within the context of the software radio. The Finite Impulse Response (FIR) filter has been used to implement a matched transmitter-receiver filter couple. However, unlike a DSP where multiply-and-accumulate (MAC) functionality is built-in, many realisation structures and data types were investigated. A summary of those findings is presented in this chapter.

Chapter 10 investigates the nature, and plausibility of implementing an MMSE multi-user detector on an FPGA. While the downlink is fairly easy to implement, especially in the light of the ideas developed in Chapter 9, many of the computational components of the uplink detector should ideally be partitioned between an FPGA and a DSP. However, since no DSP is available in the software radio used in the project, only the downlink will be investigated in this chapter. This will include a mathematical examination of the chipping sequence used, in conjunction with a fixed-coefficient MMSE detector structure.

Chapter 11 has the intention of exposing the reader to the high-speed simulating capabilities of the software radio. While some of the simulations undertaken in this chapter would consume copious amounts of time on a personal computer, the software radio demonstrates not only its flexibility, but also its speed in a simulated environment. More specifically, these simulations are set up with the intention to test the core ideas of this thesis over a simulated communications channel. This channel will have two common distortions – that of Additive White Gaussian Noise (AWGN) and uncorrelated Rayleigh fading. Issues such as speed and accuracy are explored, in conjunction with the issue of resource partitioning to optimise the load balance between the software radio and the DSP subsystem. Included in this chapter are some specific time-domain graphs showing the effect of the channel on a real signal. This chapter concludes with results obtained for system performance in both AWGN and uncorrelated Rayleigh fading channels.

This thesis concludes in Chapter 12 with suggestions for future development and improvement of the system developed in this thesis. The project undertaken in this thesis embodies the complete construction of a CDMA communications system. Each of the components has been created, and in the process, a substantial basis has been developed in the form of a library of components. This is the first step in the continuing development with the software radio.

Successive projects already have many elements in place, and these may either be simply incorporated, or they may be extended to include enhanced features. In any case, this facet of the project has highlighted one of the core aspects of software radio development: while the platform is highly flexible and allows for rapid development of new systems, the initial library of routines really has to be created from the ground up. This is primarily to accommodate the specific features of the chosen software radio platform. Thereafter, the library may be enhanced and the move towards a fully featured system is easier.

Chapter 2 Introduction to Spread Spectrum

Spread Spectrum, as the name suggests, is a technology where paradoxically, in a world of carefully regulated resources, information is spread to occupy many times the required bandwidth for wireless transmission. Spread Spectrum techniques provide the wireless communications developer with a host of exciting and powerful possibilities. For instance, it offers the prospect of its coexistence, in a near-transparent fashion, with narrow-band signals. From this, it is clear that spread spectrum signals are fairly immune to jamming signals. In this light, it may be asked why the technology has not been more widely used. The truth is that it has been used for decades in a military context, and the answer to its slow infiltration of civilian use lies in the fairly complex nature of its implementation. However, as the mobility of the world's telecommunications infrastructure has dramatically increased, so the demand for new signal transmission techniques such as spread spectrum has also amplified. This, coupled with formidable technological advances, has led to the feasibility of using spread spectrum within a civilian context.

Direct Sequence Code Division Multiple Access (DS-CDMA) is a particular type of spread spectrum communications that is gaining popularity in emerging wireless standards. Once the general philosophy of spread-spectrum communications has been established, this chapter will introduce many concepts related to DS-CDMA. Initially, this chapter will highlight some of the advantages of DS-CDMA systems over Time and Frequency Division Multiple Access (TDMA/FDMA) systems. A discussion of the processing gain introduced by the spreading codes, and the degree to which this process offers immunity to interfering signals, will help to outline the functional blocks of a DS-CDMA system. This will bring the discussion to the choice and generation of code types, especially in context of the downlink. A brief discussion of acquisition and tracking techniques will complete the theoretical component of the chapter. Thereafter, the chapter will be devoted to the incorporation of DS-CDMA in emerging standard communications protocols.

2.1 The Nature of Spread Spectrum

Spread spectrum is a technology that generally opposes the traditional approach to wireless communications in that it does not attempt to compress large amounts of information into the smallest possible bandwidth. Instead, according to [Pickholtz, Schilling & Milstein, 1982],[5], in their extensive work on outlining the theory of spread-spectrum communications, the following definition is proposed:

> *"Spread spectrum is a means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information; the band spread is accomplished by means of a code which is independent of the data, and a synchronised*

reception with the code at the receiver is used for despreading and subsequent data recovery"

This transformation of the data may, at first, seem wasteful. However, as 50 years of the use of this technology has proved, there are many advantages to its incorporation in a wireless communications system. Further, [Schilling et al, 1982], [5], identify the following as valuable characteristics of spread spectrum technology:

- Anti-jamming
- Reduced Interference
- The reduced chance of interception by the incorrect receiver
- The extensibility of random multiple user access

Some of these features will be expounded in subsequent discussion. However, it is important to first understand the mechanism for implementing this spreading philosophy, since there are three variations that have gained popularity. Possibly the most simple to conceptualise is the first system, that of direct sequence. Here, each bit to be transmitted (within a related bit period of T) is replaced with a series of bits (known as a code) that also exist within the original parameter T. This obviously generates a far higher frequency data stream, and this has the effect of spreading the single bit from its narrow-band spectrum to a spread spectrum. The second system is the frequency hopping system. In a similar fashion, the data to be transmitted is spread across a broader spectrum. However, this is not achieved by replacing the codes in the time domain. Instead, as data transmission proceeds, so the transmission frequency is "hopped" from one value to another. A third possibility is a hybrid system making use of these together in one system. The ensuing discussions will expound on the mathematics of these systems.

2.1.1 Direct Sequence Spread Spectrum System (DSSSS)

The Direct Sequence technique pervades many of the practical spread spectrum systems. In this system, the data bits to be transmitted are first multiplied with a higher frequency, binary sequence. This sequence is pseudo-random in nature, meaning that, while it does repeat with a very specific period, it approaches random noise; hence it is often referred to as a Pseudo-Noise (PN) sequence. The PN sequence is also referred to as a code, and is comprised of Non-Return to Zero (NRZ) bits, known in this dimension as chips. Literature suggests that systems employing direct sequences can do so at either the Intermediate Frequency (IF) stage, or at the base-band stage. However, most practical systems will spread the data at the base-band level, and that is the approach taken in this thesis.

$$r_i(t) = \sum_{i=1}^{N} A_i d_i(t-\tau_i) p_i(t-\tau_i) \cos(\omega_0 t + \theta_i) + n_w(t)$$ ..... **Equation 2.1**

Equation 2.1 describes a DSSSS system with the following parameters:

- $r_i(t)$ the received signal for user i
- $d_i(t)$ the bits to be transmitted by user i
- $p_i(t)$ the code sequence for user i
- A_i the amplitude of user i
- θ_i uniform random phase associated with user i
- τ_i uniform random time delay associated with user i
- $n_w(t)$ Additive White Gaussian Noise

Equation 2.1 gives the generalised equation for a DSSS system. Each user is given its own code to transmit data with. Generally, the users are asynchronous and therefore, the users' data will arrive at a receiver with some delay. Equation 2.1 forms the basis for the system model used in the thesis. However, the cosine term is not included in the system model since the signals remain at base-band frequencies within the software radio. The Direct Sequence technique utilised in this project will be expounded on later in the chapter.

2.1.2 Frequency Hopping Spread Spectrum System (FHSSS)

$$v(t) = \cos(\omega_i t + d(t)\Omega t + \theta)$$ ..... **Equation 2.2**

Equation 2.2 defines a frequency-hopped spread spectrum signal. Here, ω_i is the carrier signal that alters at a specific frequency f_H. The system will have a pre-defined set of available frequencies, and, as transmission proceeds, so the carrier frequency is chosen in a pseudo-random manner. The advantage of this system over DS systems is that, in the event of a jamming signal transmitting at a specific frequency, the FH system will transmit information in bands outside of the interfering signal. The DS system has to rely on the extent to which the base-band signal has been spread to attain successful transmission in the presence of an in-band interfering signal. Both systems rely on pseudo-random patterns for their transmission sequences. Further, to minimise Multiple Access Interference (MAI), it is important that the sequences be orthogonal.

2.1.3 Hybrid Direct Sequence/Frequency Hopping Systems

A further technique for spreading the base-band information is by combining the effects of the preceding two processes. Here, while each user bit is replaced by a higher frequency sequence, this sequence is not transmitted on a single frequency. Instead, the transmission frequency is hopped to increase the bandwidth further. This is especially useful in anti-jamming applications but is not considered further in this thesis.

2.1.4 The Orthogonal Condition

Fundamental to the successful transmission of signals in a Spread Spectrum system is the choice of the pseudo-random sequences. These codes need to be orthogonal to reduce the effect they will have on users of other codes in the system. Effectively, if there are n codes used to chip base data, then these codes must form an orthonormal basis spanning the n-dimensional space, obeying the following condition:

$$\int_0^T c_i(t)c_j(t)dt = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad \text{.....Equation 2.3}$$

A further important consideration is the power associated with each code. From the receiver's perspective, it is assumed in Equation 2.3 that the occurrence of the codes is derived from a uniform random distribution, and the energy associated with each code is constant. This introduces a further concept, that of the near-far effect. Here, a nearby interferer may overpower the signal transmitted by a distant user. This effect is overcome using power control, in which the power level of the transmitted signals is dynamically updated in an attempt to preserve equal received signal power at the receiver (or base station). This process of power control has not been implemented in the system, and remains the work of future investigations.

2.1.5 Processing Gain and Anti-Jamming Features

Herein lies one of the greatest benefits of a system utilising CDMA as opposed to TDMA or FDMA. The processing gain of the system is defined as the relative Signal to Noise Ratio (SNR) of the spread signal to the base-band signal. The relative immunity of a spread spectrum signal to a jamming (or interfering) signal is then revealed in the following equation:

$$G_P = \frac{T_d}{T_c} = \frac{R_c}{R_d} \qquad \text{.....Equation 2.4}$$

- T_d Period of base-band data bit
- T_c Period of direct-sequence chip
- R_c Chipping rate
- R_d Base-band data rate

[5] clearly shows the impact of the processing gain term in the following equation:

$$\left(\frac{E_b}{\eta_o}\right) = \frac{P_s}{P_N + P_J} \cdot \frac{B_{ss}}{R} \approx \frac{P_s}{P_N + P_J} \cdot G_P \qquad \text{.....Equation 2.5}$$

$$\therefore P_{J\max} \le \frac{P_S}{\left(\frac{E_b}{\eta_o}\right)} \cdot G_p - P_N$$

- P_s Power in the transmitted spread spectrum signal
- P_J Power in the jamming signal

- η_o White noise power density
- P_N White noise power
- B_{SS} Spread signal bandwidth
- R Base-band data rate

The first line of equation 2.5 shows that, without the presence of a jamming signal, the

While the spreading of the signal offers no improvement in terms of noise immunity (since the noise power density is considered constant over the bandwidth of the spread signal), Equation 2.5 demonstrates the ability of the processing gain to alter the system performance in the presence of a jamming signal existing within a specific frequency band. Given that the system design might require a specific SNR - $\left(\frac{E_b}{\eta_o}\right)$ - and the data rate is set at R, then setting the noise power at P_N, there is a specific maximum P_J of a jamming signal above which the data rate will not be met. However, the power of a DSSS system lies in its ability to adapt under these conditions. If a jamming signal does exist, then the processing gain simply needs to be increased, thereby allowing the spread signal to be received at the specific SNR and the correct data rate. Since it is clear that the strength of a spread spectrum system lies in spreading the base-band data with a higher dimension code, the design of these codes becomes imperative to the system's performance.

Matched Filter Reception

Many current, practical CDMA systems make use of matched filters for reception. This is a fairly simple process whereby the received signal, after it has been down-converted to base-band, is correlated with a specific code:

$$R_C(\tau) = \frac{1}{L}\sum_{i=1}^{L} C_{Ai} \cdot C_{Bi\tau} \qquad \text{.....Equation 2.6}$$

- C_{Ai} Bit **i** of Code A
- $C_{Bi\tau}$ Bit j of Code B, with associated time delay
- L The length of the code in chips

Equation 2.6 can also be applied in the case where A = B, in which case the autocorrelation is found. Generally, it places two requirements on the codes and their associated correlations

- The autocorrelation function must have a sharp peak for $\tau = 0$, while producing a very low output for all other values. This allows for correlation synchronisation
- The cross-correlation function must have low output values for all τ if A is not equal to B. These requirements are met in several classes of random, or pseudo-random, sequences.

2.2 Pseudo-Random Sequences

There are many families of codes available to the CDMA designer. However, as [Shilling et al, 1982] point out in [5], there are 4 properties which are desirable in all PN-sequences:

- They are easily generated
- They are highly random
- They have long periods
- They are not easily reconstructed given a short sample of the code bits

Some of these code systems that will be briefly examined here include maximal-length sequences, Gold codes and Walsh-Hadamard codes.

2.2.1 Maximal Length Codes

This subset of codes is defined in terms of its generation technique. Making use of Linear Feedback Shift Registers, a binary code sequence can be generated from the following equation:

$$C_n = \left(\sum_{k=1}^{r} a_k C_{n-k} \right) \bmod 2 \qquad \text{.....Equation 2.7}$$

Equation 2.7 is represented in Figure 2.1.

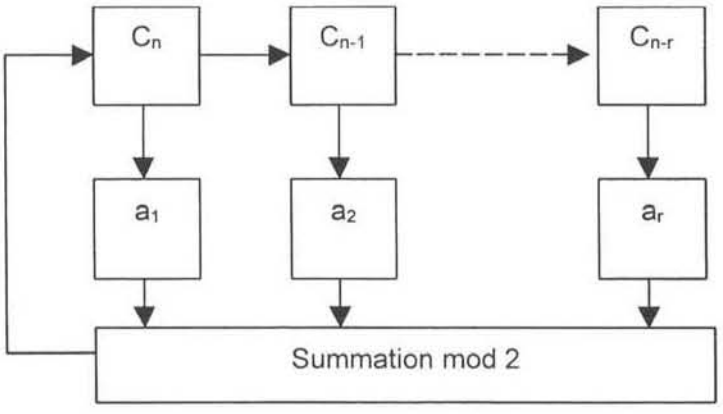

Figure 2.1 – Linear Feedback Shift Register for Pseudo-Random Sequence Generation

If the sequence is maximal length, it will produce a stream of bits C_n, where the period of the sequence is $2^r - 1$. The sequence will be maximal length if the feedback weights are appropriately designed, and the initial register state is non-zero. One characteristic requiring attention is that of consecutive runs. Clearly, the maximum spreading effect in a DSSS system is reached when there are minimal consecutive ones and zeros. For instance, a code comprised entirely of ones performs no spreading gain at all. In this light, maximal length sequences have the following properties:

- There are approximately an equal number of zeroes and ones within the sequence period
-

$$n = \frac{1}{2^l} \qquad \text{.....Equation 2.8}$$

  - n is the number of runs of consecutive ones or zeroes of a particular length
  - l is the particular length of the run of consecutive ones or zeroes

So, half of the runs of consecutive ones or zeroes will be of length 1, a quarter will be of length 2 etc.

- The sequence generated by Equation 2.7 can be modified to be a Non-Return to Zero (NRZ) stream by setting

$$C_n' = 2.C_n - 1 \qquad \text{.....Equation 2.9}$$

Under these conditions, the autocorrelation function will produce the following equation

$$RC'(\tau) = \begin{cases} 1 & \tau = n.L, n \in Z \\ \dfrac{-1}{L} & otherwise \end{cases} \qquad \text{.....Equation 2.10}$$

Equation 2.10 is shown in the autocorrelation plot for the following maximal length sequence where r = 8 and the taps occur at $n \in (8,4,3,2)$.

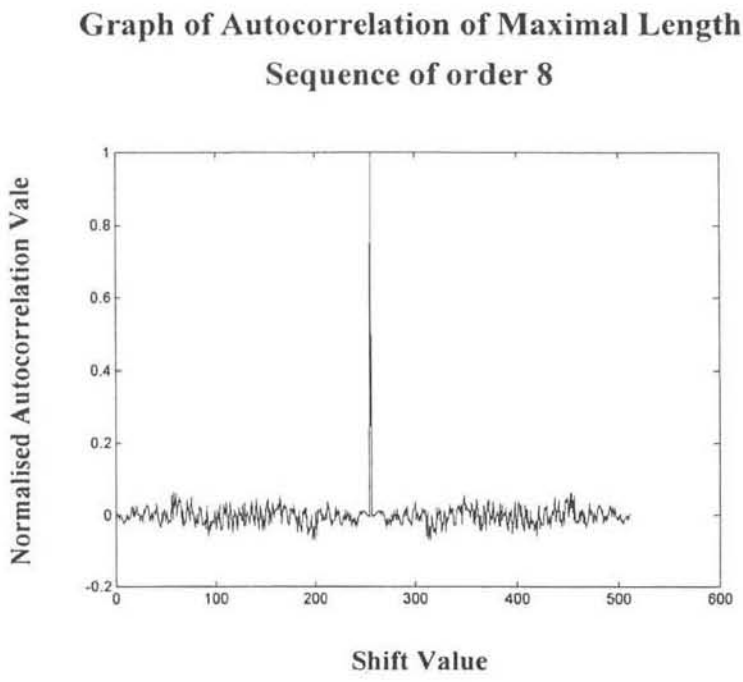

Figure 2.2 – Autocorrelation of a Maximal Length Sequence of Order 8

If the autocorrelation peak is excluded from the calculations, it is also true that the mean of the autocorrelation function is $-1/2^{r+1}$. Further, the cross-correlation mean for two independently generated maximal length PN sequences is zero. From this, it is clear that the correlation process is key in terms of synchronising to the transmitted code, since the autocorrelation peak is highly defined when compared with the rest of the autocorrelation values. Correlating detectors can therefore "scan" for the correct synchronisation by shifting the reception code until the sharp autocorrelation peak is found. Then, the receiver should be synchronised to the transmitted code. Also important, in this technique, is the corresponding cross-correlation peak value. If this is significant, it may be confused with the autocorrelation peak and incorrect synchronisation may ensue. For this reason, it is important to choose codes that have a sharp autocorrelation peak with suppressed cross-correlation values.

A downfall of the ML-sequence is the ease with which the sequence can be reconstructed after interception of a small number of bits in the sequence. In fact, [Shilling et al, 1982] reveal that only 2r – 2 bits need to be observed for an interceptor to solve, using simultaneous equations, the remaining bits of the sequence. Having said this, there are means of modifying the sequences to produce derivatives that satisfy all of the requirements of a PN sequence.

2.2.2 Gold Codes

Making use of two maximal length sequences, it is possible to derive a third sequence, which, if the constituent codes are "preferred", is termed a Gold code (refer to appendix A.3). The third sequence is generated by taking the bit-wise exclusive OR of the two maximal length sequences. However, there is a specific subset of these resulting codes that have the following cross-correlation values:

$$\{-1,-t(r),t(r)-2\} \qquad \text{.....Equation 2.11}$$

$$t(r)=\begin{cases}2^{(r+1)/2}+1 & odd\ r\\ 2^{(r+2)/2}+1 & even\ r\end{cases}$$

In Equation 2.11, r is the number of elements in the maximal length shift register. The technique for finding the preferred pairs involves a process known as decimation.

Decimation

Order q decimation of a binary sequence is the process of constructing a second binary sequence using every q^{th} sample of the first. This process is fundamental in the construction of the second sequence in the preferred pair.

$$b'=b[q] \qquad \text{.....Equation 2.12}$$

In Equation 2.12, b is the primary m-sequence, while b' is the decimated sequence. Under certain conditions, b' is also a maximal length sequence of the same order (r) as b. The following conditions are sufficient in defining a preferred pair b and b':

- $r \bmod 4 \neq 0$
- $b'=b[q]$, q is odd and either

  $q=2^k+1$

  or

  $q=2^{2k}-2^k+1$
- The greatest common divisor of N (the period of the sequence) and k (see above) is either 1 or 2.

The fundamental technique for the generation of Gold codes is well suited to a hardware environment. This is because, as seen, Gold codes are derived from maximal length sequences,

and maximal length sequences are easily generated using shift-registers. Once two preferred maximal length sequences have been identified, these two generators are used to form the Gold code set in the following manner: the first generator is loaded with a constant non-zero seed. The seed of the second generator is successively changed from a value of 0 to $2^r - 1$, producing $2^r - 1$ different maximal length sequences. On combining these sequences with those produced by the first generator, $2^r - 1$ Gold codes are generated. The ease of interception of a Gold code sequence is far lower than that of a maximal length sequence. However, the major advantage of Gold codes over maximal length sequences is seen in an examination of the cross-correlation features. [Xsiology], a wireless Internet company, in their document discussing code systems, [8], reveals the power of Gold codes in Table 2.1.

Table 2.1 – Cross-correlation Comparison between M-Sequences and Gold Codes

Register length r	**Code length l**	**Number of maximal length sequences N**	**Peak cross-correlation value for maximal length sequences P_M**	**Normalised cross-correlation peak P_M/l**	**Peak cross-correlation value for Gold code sequences P_G**	**Normalised cross-correlation peak P_G/l**
3	7	2	5	0.71	5	0.71
4	15	2	9	0.6	9	0.6
5	31	6	11	0.35	9	0.29
6	63	6	23	0.37	17	0.27
7	127	18	41	0.32	17	0.13
8	255	16	95	0.37	33	0.13
9	511	48	113	0.22	33	0.06
10	1023	60	383	0.37	65	0.06
11	2047	176	287	0.14	65	0.03
12	4095	144	1407	0.34	129	0.03

As can be seen in Table 2.1, Gold codes have superior cross-correlation features compared to maximal length sequences. The normalised, maximum cross-correlation peak associated with maximal length sequences is far higher than with Gold codes. The result is that, with maximal length sequences, when autocorrelation is used to synchronise to a specific code, the corresponding synchronised peak may be confused with a peak generated by the cross-correlation with a different code. This is, to a large extent, avoided when Gold codes are used. However, Gold codes are not truly orthogonal, and therefore may cause unacceptably high levels of multiple access interference.

2.2.3 Walsh-Hadamard Codes

Walsh sequences are not constructed from shift register sequences (unlike m-sequences, gold codes and Kasami codes). They are mathematically developed to form an orthonormal set – ideal in minimising multiple user interference. While there are several methods of deriving this orthonormal set, the Hadamard matrix method is the easiest to implement under simulation conditions. Here, there are matrices of varying order, whose rows or columns represent the orthogonal PN sequences, all derived from the Hadamard matrix of lowest order:

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \text{.....Equation 2.13}$$

The iterative derivation of the various matrices is as follows:

$$\mathbf{H}_N = \mathbf{H}_{N/2} \otimes \mathbf{H}_2 \qquad \text{.....Equation 2.14}$$

This equation makes use of the direct (Kronecker) matrix multiplication. Please refer to A.1 in the appendix for the mathematical properties of such a product. The orthogonality of the codes is seen in the fact that:

$$\sum_{t=0}^{N-1} WAL(m,t)WAL(n,t) = \begin{cases} N & n = m \\ 0 & n \neq m \end{cases} \qquad \text{.....Equation 2.15}$$

WAL(n,t) is the generating function for an ordered set of Walsh functions. Here, the functions are defined over a specific interval T. In order to define the function completely, two parameters are required

- t This is the time period, often normalised to the base-time T
- n The ordering number, related to the frequency of the generated Walsh function

Stated in terms of continued products, the constituent function of Equation 2.15 may be written as

$$WAL(n_{p-1},n_{p-2},\ldots,n_0;t_{p-1},t_{p-2},\ldots,t_0)=\prod_{r=0}^{p-1}(-1)^{n_{p-1-r}(t_r+t_{r+1})} \quad \text{.....Equation 2.16}$$

Here, n and **t** are presented in binary notation, while $N = 2^p$ is the desired number of terms in the Walsh sequence. While these codes are truly orthogonal, they do have some practical limitations:

- Chip-level synchronisation is problematic due to the lack of a single sharp auto-correlation peak.
- When the PN spectrum is analysed, it is apparent that the spreading energy is located at specific frequencies, instead of being spread over the entire spectrum (as is the case with shift register sequences). This is due to excessive consecutive runs of 1's and 0's.
- Partial correlation is highly problematic for Walsh sequences, since the codes are non-orthogonal under these conditions:

Table 2.2 – Cross-correlations for Walsh Codes

Walsh Order N	Peak Cross-Correlation	Mean Cross-Correlation	Mean RMS Cross-Correlation
2	0.5	0	0.125
4	0.75	0	0.219
8	0.875	0	0.289
16	0.9375	0	0.342
32	0.9688	0	0.381
64	0.9844	0	0.411

As can be seen in Table 2.2, the cross-correlation performance of Walsh codes means that they are unacceptable for use in asynchronous correlation detection techniques. However, in synchronous networks, the cross-correlation of Walsh codes is exactly 0, as seen in Equation 2.15.

2.3 Asynchronous CDMA

Clearly, from Equation 2.1, there is a time delay associated with each user at the receiver of several superposed CDMA signals. This presents a problem for the multi-user detection schemes presented in subsequent chapters where it is required that either the users are truly synchronous in the network, or their relative delays can be estimated. This section examines the latter case where algorithms may be employed to estimate the delays associated with each user. Many algorithms exist for estimating these delays. However, it is also true that many of these require a training sequence of known transmitted user bits before the delay can be

accurately estimated. Since it is undesirable to have to such a training sequence during tracking, the algorithm needs to operate independently of such data. Strom et al report in [29] that the Multiple Signal Classification (MUSIC) algorithm is capable of generating delay estimates without any knowledge of the data sequences being transmitted. The basic MUSIC algorithm is presented here.

2.3.1 The MUSIC Algorithm

The Multiple Signal Classification (MUSIC) algorithm is often used as a synchronisation technique in CDMA systems. According to Harri Saarnisaari in [28], one of the reasons for this choice is that, given that the number of interfering users is less than a third the length of the system codes, the algorithm is actually near-far resistant. Consider the following simpler representation of Equation 2.1, where the received signal is represented by N samples in the k^{th} observation period:

$$\mathbf{r}_k = a(k)\mathbf{s}(\tau_0) + \mathbf{i}_k + \mathbf{n}_k \qquad \text{.....Equation 2.17}$$

Where the following variable definitions apply:

- $\mathbf{r}_k$ The sample vector at the k^{th} observation interval
- $\mathbf{s}(\tau_0)$ The sampled desired signal with unknown delay τ_0
- a(k) The associated complex amplitude
- $\mathbf{i}_k$ Interference vector
- $\mathbf{n}_k$ AWGN vector

Suppose that the interference may be modelled by N-1 sources in the following manner:

$$\mathbf{i}_k = \sum_{n=1}^{N-1} a_n(k)\mathbf{s}_n \qquad \text{.....Equation 2.18}$$

Suppose also that the sample correlation matrix can be found such that:

$$\mathbf{G} = \frac{1}{K_O}\sum_{k=1}^{K_O} \mathbf{r}_k \mathbf{r}_k{}^H \qquad \text{.....Equation 2.19}$$

Here, the sample correlation matrix **G** is based on an observation period of length K_O – the algorithm performs accurately only when K_O is very large. Suppose now that the correlation

matrix is decomposed to its eigenvalues, λ_i, with an associated eigenvector $\mathbf{e}_i$. Then, according to H. Saarnisaari in [28], the MUSIC estimator simply maximises the following:

$$f(\tau) = \sum_{i=1}^{p} \left| \langle \mathbf{e}_i, \mathbf{s}(\tau) \rangle \right|^2 \qquad \text{.....Equation 2.20}$$

In Equation 2.20, $\langle \mathbf{x}, \mathbf{y} \rangle$ is the inner product, $\mathbf{y}^H\mathbf{x}$ while p is the total number of desired and interfering signals. For optimal performance, it is also necessary that the parameter p be accurately estimated.

MUSIC Algorithm Performance

The performance of the MUSIC algorithm under various channel conditions needs to be examined. If the source of the interference described previously is due to multi-path propagation, then it is preferable that the received amplitudes are uncorrelated. However, if they are fully correlated, the MUSIC algorithm will fail to resolve finely spaced timing delays. If the source of interference is not due to multi-path propagation, then it is only required that the amplitude of the desired signal not be fully correlated with the amplitudes of the interfering signals. T. Ostman and B. Ottersten, in [30] discuss the application of the MUSIC algorithm to systems with band limited pulse shapes. They conclude that the "behaviour of the MUSIC algorithm is very similar when rectangular pulses and when square-root raised cosine pulses are used". This thesis examines the case where the transmitter and receiver filters are both square-root raised cosine transfer functions. Therefore, the MUSIC algorithm is applicable to the system presented in this thesis. However, due to the computational complexity of the MUSIC algorithm and other delay estimation algorithms for DS-CDMA systems, no such algorithm has been incorporated into the practical implementation of the CDMA system in this thesis.

2.4 Conclusion

This chapter has examined many of the mathematical issues related to spread spectrum communications. Three key issues have been examined to formulate a firm context for spread spectrum. First, its many idiosyncrasies and hybrid implementations were established as its nature was expounded. Focussing on direct sequence CDMA as the spread spectrum technique of choice, the second issue, that of orthogonal pseudo-random sequences, was examined. The importance of designing these codes to adhere to stringent orthogonal and partial cross-correlation levels has seen several families emerge in strong contention for use in DS-CDMA systems. Finally, the third issue was an introduction to the problems associated with asynchronous CDMA systems. The MUSIC algorithm was presented as one possible means of

estimating timing delays between users in a CDMA system. Due to its complexity, however, it has not been implemented in this project. Chapter 3 will examine a new technique for detecting the information transmitted by several users, each making use of their own unique code.

Chapter 3 Introduction to Multi-User Detection

Multi-User Detection is a progression in CDMA detection techniques towards the simultaneous demodulation of all data streams present in a single multiple-access channel. This technique is vastly different to that suggested in the previous chapter, where the correlation procedure was key in detecting the transmitted data of a single user. Under a correlating system, the receiver will attempt to extract this data stream from the received signal. It does this by treating the other users' signals as Additive White Gaussian Noise, while correlating the received signal with the specific code used by the transmitter of interest. If a correlation peak is obtained, it is assumed that the transmitter and receiver are synchronised. However, if this peak is not seen, then the receiver introduces successive time delays until the peak is found. From this, it follows that the performance of such a receiver is tightly integrated with the choice of appropriate codes. These codes are capable of producing very low cross-correlation values (as shown in the previous chapter), but they may necessitate synchronisation (as in the case of Walsh codes) and limit the number of users. In spite of the utmost care in the design and choice of the system codes, they still cannot overcome the physical defects associated with the communications channel - as with narrowband signals, spread spectrum signals still suffer from transmission errors due to AWGN, multi-path propagation and shadowing. However, the primary downfall of this correlation process is the manner in which the signals generated by other users are treated. These signals cause Multiple-Access Interference (MAI), a phenomenon that the filter will treat as Additive White Gaussian Noise. This assumption is invalid and, since the interference is highly structured, it is actually wasteful to discard this information. Taking these arguments related to the correlating detector into consideration, it appears that an improved detector would incorporate the following features:

- The receiver will not only process the entire received waveform, but will be aware of other codes being used, and will be able to use this information to produce a better replica of the transmitted data
- Such a receiver should also be aware of the channel conditions, and use this information to further improve its performance.

A receiver incorporating these attributes is known as a multi-user detector. Since it makes use of the code information transmitted by other users, it is also capable of simultaneously detecting all of the other streams transmitted on a single channel. Also, if the channel noise can be accurately estimated, the detector is capable of suppressing distortion, thereby improving its performance. There are several types of multi-user detectors. These include De-correlating Receivers, Interference Cancellers, and Minimum Mean Square Error (MMSE) Detectors. This chapter will model the CDMA channel, developing the mathematics of an MMSE multi-user detector. It will be seen that, while promising, such a receiver has many complexities which

need to be overcome, and two solutions are outlined involving iterative and adaptive techniques. Further, the importance of Walsh codes is examined in the context of the multi-user detector, since making use of these codes produces a particularly elegant solution.

3.1 The System Model

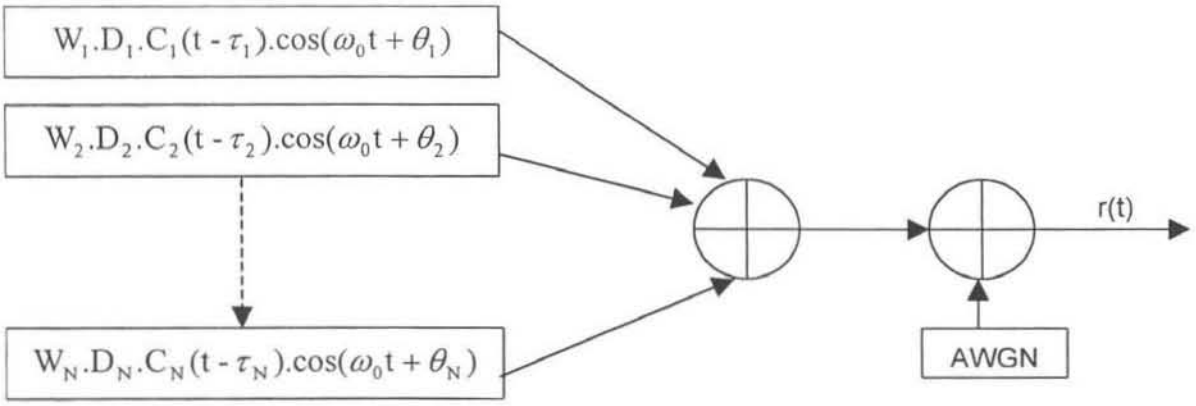

Figure 3.1 – The Asynchronous CDMA Channel Model

Figure 3.1 shows a typical asynchronous CDMA channel model, where each user is parameterised in the following manner:

- W_i The relative amplitude of user i
- D_i The data source for user i
- C_i The code used to spread the data for user i

3.2 System Mathematics

It is easiest to formulate r(t) in terms of the above by making use of the following matrices:

- **A** The (NxK) code matrix (real valued for synchronous case, complex-valued for asynchronous case) where each column i contains the unit energy code for user i (N codes used by K transmitters)
- **W** The (KxK) diagonal weighting matrix where element (i,i) is the relative transmission amplitude for user i
- **d** The column vector (Kx1) with the current data element for each user
- **n** The column vector (Nx1) containing sampled complex noise

In light of the above, the following equations are applicable:

$$E[\mathbf{nn}^*] = \sigma^2 \mathbf{I}_K \qquad \text{.....Equation 3.1}$$

Here, σ^2 is the sampled variance of the complex noise, while $\mathbf{I}_K$ is the (KxK) identity matrix. Further, the received signal may be concisely written as follows:

$$\mathbf{r} = \mathbf{AWd} + \mathbf{n} \qquad \text{.....Equation 3.2}$$

As already stated, this chapter will examine the MMSE variation of the multi-user detector. The objective with the MMSE is to minimise the following Euclidian distance:

$$\|\mathbf{r} - \mathbf{AWd}\|^2 \qquad \text{.....Equation 3.3}$$

In the case of the MMSE multi-user detector, a filtering matrix **H** is derived which, when multiplied with the received data vector, will produce a stream of all the current user's data. [Grant & Sclegl, 1999] show in their work on iterative techniques for multi-user detectors, [10], that this filtering matrix is

$$\mathbf{H} = (\mathbf{WRW} + \sigma^2\mathbf{I})^{-1}\mathbf{WA}^* \qquad \text{.....Equation 3.4}$$
$$(\mathbf{R} = \mathbf{A}^*.\mathbf{A})$$

R is made up of the normalised correlation matrix, taken between the codes of the users in the system, while **A*** is the Hermitian (complex) transpose of the code matrix **A**. Equation 3.4 shows that, given prior knowledge of the code database, it is possible to generate a matrix **H** which can extract all of the users' data simultaneously.

$$\mathbf{d'} = \mathbf{H.r} \qquad \text{.....Equation 3.5}$$

Equation 3.5 shows that, through a simple matrix multiplication, a data estimate may be generated for every user registered in the system. However, Equation 3.4 is difficult to implement because it contains a matrix inversion. There are three approaches to overcoming this bottleneck from an implementation perspective. First, the developer may choose to have an adaptive implementation that "learns" the vital characteristics of the transmission (which codes have been used and what the characteristics of the channel are). Second, the developer can attempt an iterative solution of the matrix inversion that may prove slow. Finally, there is the option, with Walsh codes, of using fixed coefficients where the noise acts in a highly predictable manner on the filtering matrix **H**.

3.3 Iterative Solutions

The bottleneck in the MMSE filter, in its direct form, is clearly the matrix inversion operation.

$$\mathbf{Mx} = \mathbf{b} \qquad \text{.....Equation 3.6}$$

Equation 3.6 is an alternate means of stating the matrix inversion operation – it shows the situation applied to any (nxn) matrix **M**, and its associated n-length column vector **b**, and **x** is the vector solution to the problem. Solving for **x** directly will require $O(n^3/3)$ operations. However, there are iterative means of solving for **x** that allow for convergence in fewer operations. In their paper on the subject, [Grant & Schlegel, 1999], [10] reveal several techniques for iteratively solving the matrix inversion in Equation 3.4. The following discussion incorporates and tests many of their ideas.

Splitting the Matrix

The matrix **M** may be split into two constituent matrices **S** and **T** such that **M** = **S** – **T**, and Equation 3.6 may be therefore re-written, in iterative form, as:

$$\mathbf{S}x_{i+1} = \mathbf{T}x_i + \mathbf{b} \qquad \text{.....Equation 3.7}$$

For the MMSE multi-user detection process, the following two equations apply:

- $\mathbf{M} = \mathbf{W}.\mathbf{R}.\mathbf{W} + \sigma^2\mathbf{I}$ and
- $\mathbf{b} = \mathbf{W}.\mathbf{A}^*.\mathbf{r}$.

There are several iterative techniques available to the developer, and each is associated with its own means of splitting **M** into the constituent **T** and **S** matrices. Three popular techniques are:

- Jacobi sets **S** to be the diagonal part of the matrix **M**
- Gauss-Seidel sets **S** to be the summation of the diagonal part and the strictly lower part of the matrix **M**
- Successive relaxation:

$$\mathbf{S} = \mathbf{D} + \omega\mathbf{L} \qquad \text{.....Equation 3.8}$$

Where **D** is the diagonal part of **M** and **L** is the strictly lower part of **M**.

Results

Appendix A.2 shows that there is an iteration point beyond which it is more efficient to perform a direct matrix inversion. For instance, if two non-preferred maximal length sequences are chosen to generate a pseudo-gold code matrix, then Figure 3.2 can be expected.

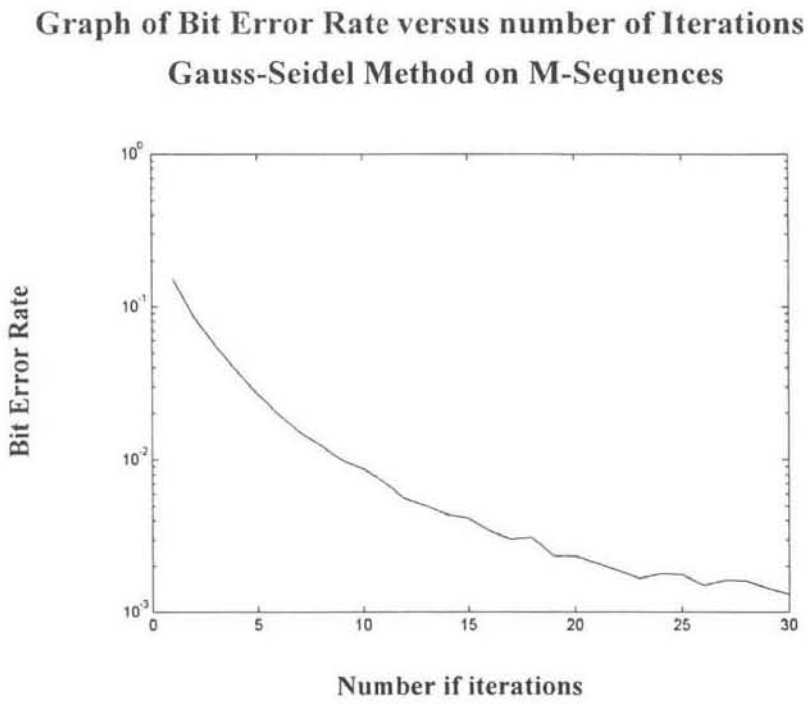

Figure 3.2 – The Gauss-Seidel Method Applied to M-Sequences

Appendix A.2 places a limit on the efficient number of iterations, a value that approaches 5. It is obvious from Figure 3.2 that, at 5 or less iterations, the system still has an unacceptably high bit error rate. This is due to the poor cross-correlation properties of the maximal length sequence codes. On the other hand, if Walsh codes or Gold codes are used, it is possible to achieve a 0 BER in a single iteration if the initial guess of the data is a vector containing only 1's (without the presence of noise). This is true for both Jacobi and Gauss-Seidel techniques. It is also of interest to see how these two sets of codes perform in the presence of noise using the MMSE filter with iterative matrix inversion

Iteration Success with AWGN

Since both Gold codes and Walsh codes converge on the correct solution within a single iteration, it is of value to observe the performance of the iterative solution in the presence of AWGN. This has been performed for 100000 bits for each Signal to Noise Ratio, both with Gold codes and Walsh codes.

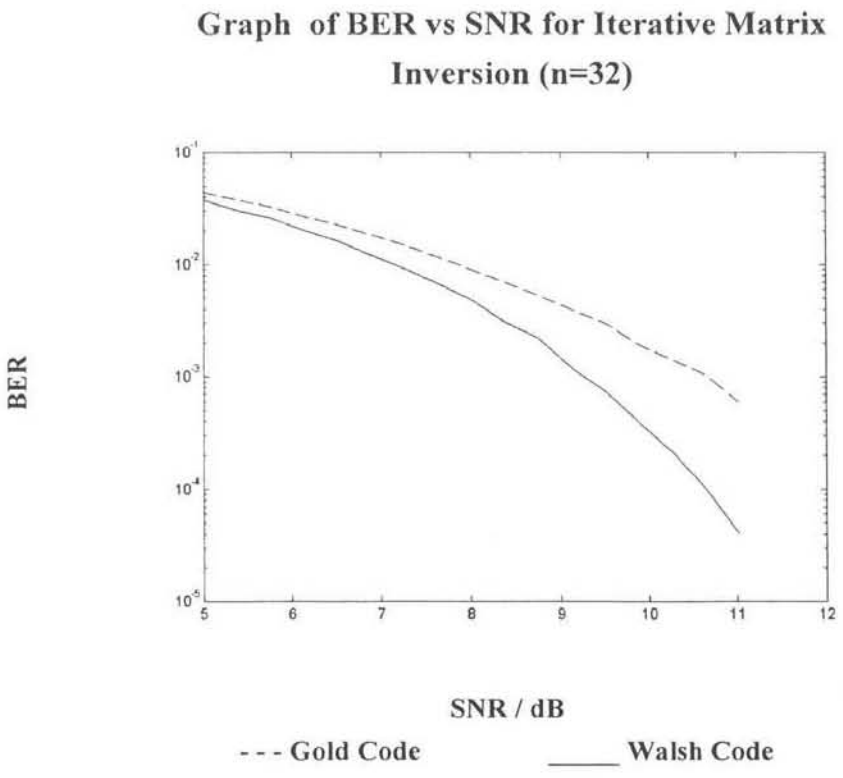

Figure 3.3 – Comparison of BER between Gold Codes and Walsh Codes – Iterative Scheme

Figure 3.3 shows that, in the iterative case, Walsh codes outperform Gold codes in noise performance testing. However, both perform satisfactorily for their relevant signal to noise ratio levels.

3.4 Fixed Coefficient Solutions – The Walsh Code Case

A CDMA system making use of Walsh codes in a MMSE detector can take advantage of two properties of the code database. Suppose that there are N codes available for use, each being stored in the column of a matrix **A**. Then, without noise, the calculation of **H**, the MMSE filter matrix, reduces to a very simple format.

$$\mathbf{H} = \frac{1}{N} \cdot \mathbf{A} \qquad \text{.....Equation 3.9}$$

This equation was originally derived from observation, but it has a sound mathematical basis. First, it is important to note that the Hermitian transpose of **A** is exactly identical to **A**, given that **A** is a real Walsh matrix. Therefore, the correlation matrix **R** is simply **A.A**. Since different columns and rows in **A** have exactly zero cross-correlation, it is also true that

$$\mathbf{R} = N.\mathbf{I} \qquad \text{.....Equation 3.10}$$

Thus, for equal user amplitudes, Equation 3.4 reduces to:

$$\mathbf{H} = \mathbf{R}^{-1}.\mathbf{A} \qquad \text{.....Equation 3.11}$$

From Equation 3.10, it is clear that **R** is easily invertible, allowing for the natural derivation of Equation 3.9 from Equation 3.11. Further, Equation 3.9 shows that, to extract a specific user requires only 2n multiplications and n additions. In the presence of AWGN, there is another means of simplifying the calculation of **H**, which also avoids matrix inversion.

$$\mathbf{H} = \frac{1}{N} \cdot \mathbf{A}\left(1 - sign(\mathbf{A}) \cdot \frac{\sigma^2}{N}\right) \qquad \text{.....Equation 3.12}$$

shows that once the noise has been estimated, a constant value of $\frac{\sigma^2}{N}$ can be added to each element of **A** (with sign adjustment as shown) for the successful cancellation of noise. In the case of a practical system, the noise estimate may be adjusted adaptively until the desired signal is successfully decoded.

The final issue to examine is that of the weighting matrix. This matrix (**W**) determines the relative amplitudes of the received signals with respect to their transmitted value. This diagonal matrix will be exactly equal to the identity matrix a non-lossy transmission channel. However, since the signals will be attenuated, matrix **W** can be used to compensate. If user i experiences an attenuation of 1/x, in the case of Walsh codes, row i of matrix **H** is multiplied by x. This is only true for the case where there is no noise present in the channel.

Once again, these desirable properties of the Walsh codes stem from their very low in-phase cross-correlation values. This technique produces a fixed-coefficient solution, and is especially suited to implementation on an FPGA.

3.5 Adaptive Techniques with the MMSE

In the previous case of the MMSE detector, it was required that several parameters of the system be know at the receiver, including signatures and timing of all of the relevant users. However, it is often undesirable to synchronise the users and share the code information. One solution to this problem is to implement an adaptive MMSE receiver. This type of receiver has not been implemented in this system, but it affords an interesting avenue of further research on

the software radios. The adaptive detector consists of an adaptive fractionally spaced digital filter [6]. There are two states in an adaptive filter – the training state and the transmission state. The disadvantage of the adaptive filter is that it does require an initial training sequence. This is a symbol sequence which is known to the receiver, and which is transmitted by a specific user on a specific code. This training sequence allows the receiver to establish the timing and filter parameters for successful reception. Thereafter, the filter will be in transmission state. In this state, the update of the filter coefficients is decision directed. The adaptive algorithm adheres to the following tap update mathematics:

$$\mathbf{v}_{n+1} = \mathbf{v}_n - \alpha \varepsilon_n \mathbf{t}_n \qquad \text{.....Equation 3.13}$$

Equation 3.13 variables:

- $\mathbf{V}_n$ is the complex tap vector at the n^{th} iteration
- ε_n is the error of stage n
- $\mathbf{t}_n$ is the delay line sample vector at stage n
- α is a small positive number. This is adjusted to affect the convergence rate

3.6 Conclusion

This chapter has briefly examined some of the issues related to simultaneously de-spreading the information of multiple users on a single CMDA channel. This exciting technology has limitations, primarily related to computational burden. However, this chapter has also examined means of reducing this load through iterative and adaptive techniques. This chapter marks the end of the fundamental issues to be developed in the thesis. Successive chapters will begin to explore many practical issues, especially those related to implementing the fundamental ideas expounded in previous chapters.

Chapter 4 The Software Radio Platform

The software radio provides the radio developer with a highly flexible tool for implementing and debugging radio designs in a modular fashion. In definition, it is a radio where many of the functions can be implemented in a re-configurable processing block. This block may be a single device or an array of similar devices working together to produce a repository of interconnected radio functions. As a concept, the software radio unites the worlds of software development and hardware implementation in an attempt to seamlessly integrate flexibility and high-power use, hiding the low-level hardware issues from the designer. In reality, the developer needs to be constantly aware that the join is as seamless as the possessed knowledge of both the related hardware and software issues. However, the advantages of the flexible radio platform are sufficient to warrant an investigation into this attractive device, discovering not only the need for such a tool, but the successful meeting of this need in the characteristics of the software radio. This chapter's purpose is then three-fold. First, the need for such a device has to be established. This need must be placed not only in a research context, but also in an industrial context, because this is the primary driving force for the research component. Second, a suitable solution must be found. At this point, it must be noted that the variations in software radio implementations are still largely a function of the meeting of some underlying project specifications, and not the outcome of a detailed and totally flexible standard. To this end, the third purpose will be tackled – that of outlining the specifications of the software radio platform used in this project – the Alcatel Altech Telecoms flexible radio platform.

4.1 Establishing the Need for a Flexible Platform

The software radio meets very real needs for both the researcher and the industrial designer. The needs identified below constitute some of the issues relating to members of both of these fields:

- Reliability and reproducibility are called into question in the analogue domain where no two components perform in an identical fashion. This has an impact not only on research efforts, where two platforms may perform differently, but also on an industrial scale where quality control and testing attains paramount importance.
- The diversity of communication standards means that one fixed, analogue platform may be ideal for research in a particular area, but its inflexibility will prevent its successful use in a field utilising a different algorithm. This is further exacerbated when the researcher wants to test a newly developed protocol but cannot afford to construct a suitable test bed.

Typically, radio design requires an in-depth knowledge of several key issues. This is generally the product not only of accumulated experience, but also of continuous research regarding the

latest developments. Further, the interfacing of system components is generally a stumbling block, especially when the design is physically implemented. However, if these processes and system components can be successfully described and shifted to the digital domain, then the concept becomes the key issue, while implementation loses its proportional importance.

Although the following are not all problems associated with older radio design techniques, there are several limitations that the software radio does overcome, including:

- The complexity of designs incorporating adaptive entities and cutting edge technology has prevented them from being successfully researched in traditional test-beds.
- Since the majority of the radio functions are found in a single processing block, the need for large numbers of discrete devices can be avoided.
- The flexibility of the software radio extends to the area of field debugging and – enhancement. Due to its re-configurable nature, the software radio may be re-programmed in the field to include enhancements and patches. Furthermore, if sufficiently flexible, these upgrades may also be performed over the air – with the service provider updating mobiles without the user being aware.
- The re-configurable nature of the software radio allows the concept to find a natural home in the internationally mobile radio. Negotiating the plethora of mobile standards becomes easier if the device can recognise a new environment and configure itself from a database of stored or retrievable protocols.

The problems that the software radio promises to overcome, in addition to the features it offers, clearly establish the need for such a device. Defining such a device is the next logical step in the discussion.

4.2 Meeting the Need – Defining the Software Radio Elements

The flexibility of the software radio is derived primarily from the fact that a large percentage of the radio functions are defined at time of use, and not at time of design. It is clear, from this definition that a software radio is intrinsically re-configurable. However, the non-specific nature of this definition does not place limits on the re-configurable element of the radio, meaning that its boundary of operation is actually rather fuzzy. However, after reflecting on the issues already discussed, this re-configurable element should constitute as large a portion of the software radio as possible. The re-configurable element comes in one of three popular guises:

1. A high speed, low power digital signal processor (DSP)
2. A large field programmable gate array (FPGA)
3. A hybrid comprising elements from (1) and (2) with resource partitioning

4.2.1 Resource Partitioning

In a system using software radio type 1 or 2, resource partitioning is simplified to a matter of assigning tasks to elements in an array of either FPGA's or DSP's. However, software radio type 3 requires some form of resource partitioning to share the radio tasks. Generally, those algorithmically simpler tasks are assigned to the FPGA, where speed and timing are critical, whereas the more complex algorithms may be implemented in a high speed DSP. Within the DSP, a delay in returned information may mean a slight increase in BER, but a complete break down in the channel is not anticipated in the event of a non-real time response. For instance, offline estimation of received signal powers and matrix inversions, used in updating filter parameters, may not be critical in terms of delay in calculation. This type of operation is ideally suited to a DSP. However, chipping data, which is critically timed, cannot tolerate timing delays, and should be implemented in an FPGA.

Support Hardware

The extent of the integration of the functions generally performed by discrete components with the re-configurable element of a software radio is a measure of its flexibility. In most current software radios, the re-configurable element is supported by a host of discrete devices to complement its functionality. This is primarily because the functions are so diverse as not to be able to draw resources from a common base. However, future integration, albeit not on the basis of resource sharing, will still improve system flexibility since functionality will be able to be built into smart device interfaces, as opposed to having to interface devices using complex, fixed circuitry. Some of the functions commonly not integrated in the re-configurable element include:

- Analogue to Digital and Digital to Analogue converters
- Mixers for up and down conversion
- Support for common bus standards
- Large memory blocks
- Dedicated micro-controllers for receiving and storing configuration data

This idea of distributed functionality is demonstrated in Figure 4.1.

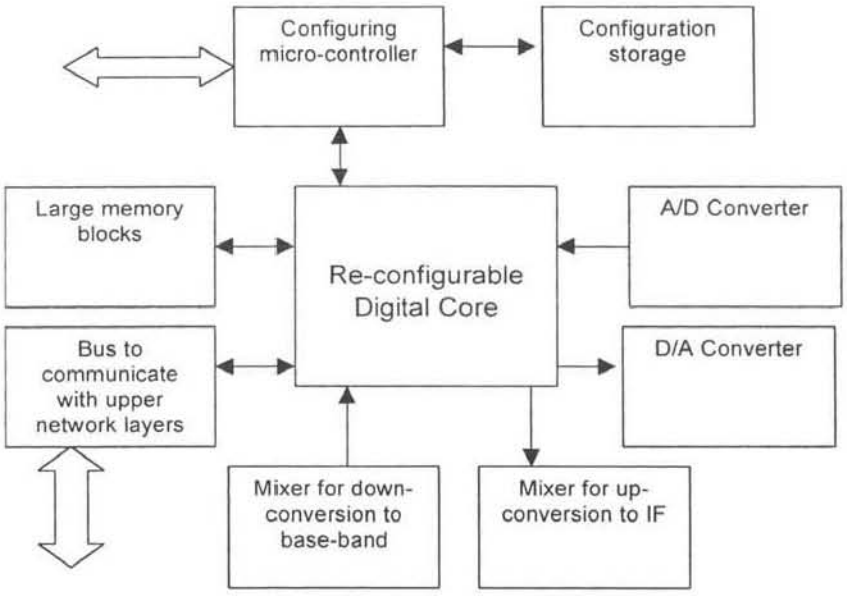

Figure 4.1 – Hardware System Blocks for a Typical Software Radio

4.3 The Alcatel Altech Telecoms Flexible Radio Platform

The technical specifications of the Alcatel Altech Telecoms flexible radio platform, found largely in [24], set the stage for several issues in the remainder of the thesis. These criteria will set limits on parameters such as data rates and mixing frequencies.

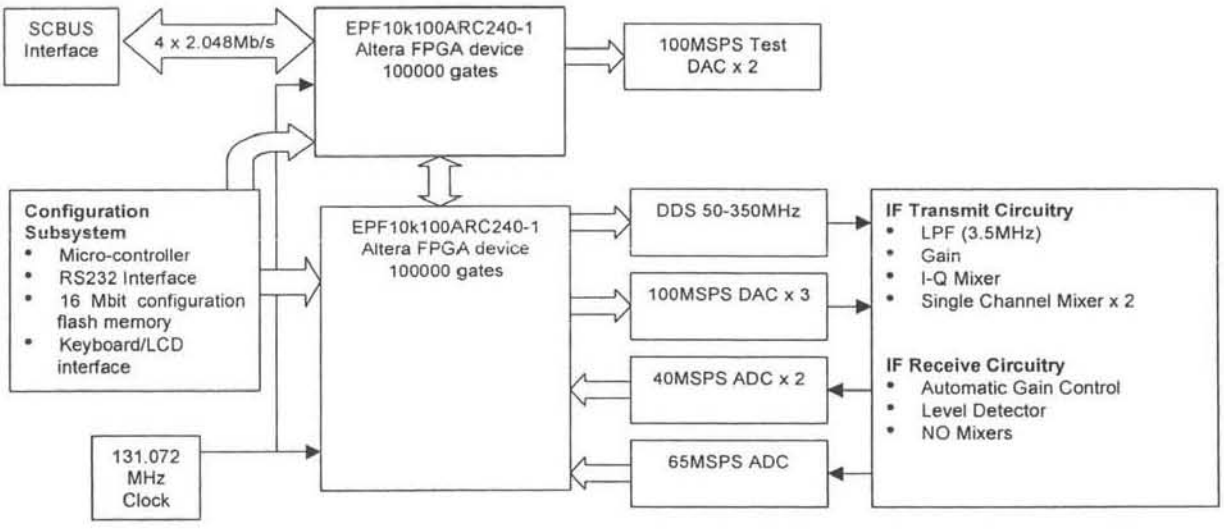

Figure 4.2 – The Alcatel Altech Telecoms Flexible Radio Platform

Figure 4.2 shows the basic inter-connection of the re-configurable component (two 100000 gate FPGA's) and the support hardware. This software radio component is designed to communicate with a DSP subsystem (based on a PCI6200 DSP card). However, the interface with the card cannot be used for high-speed parallel communications (since the bus consists of

four bi-directional 2.048 Mbps serial links). In terms of resource partitioning, the DSP system can really only act as a source of slow-changing control data and/or information data. So, as previously discussed, this system is not really capable of partitioning functions such as matrix inversion and power estimation in high mobility implementations. However, it is possible to perform these functions, although the implementation speed will be restricted by the bus speed.

The system clock speed has been carefully designed to accommodate for a large range of standard communications rates. For example, the slotted SCBUS timing, operating at 2.048 Mbps, is exactly 64 times slower than the system clock. Thus, derivation of this clock speed is simple, while the basic data rate of 32 TDMA users on the SCBUS (with 8kHz frames) is 64kbps, also an easily derived clock rate. In addition, the synthesised logic elements for the FPGA's rarely exceed a clocking speed of 100MHz, meaning that the silicon speed is also suited to this clock rate.

The up-converting mixers have their oscillators controlled by a Direct Digital Synthesiser (digital-controlled oscillator). This component can generate signals from 50MHz to 350MHz. However, the filters on the board have been optimised for 60MHz – 80MHz operation, and require hardware alterations for different frequency ranges. The hardware allows for BPSK and QPSK modulation. However, the system has been designed with no down-converting mixers. This presents a difficulty, since the demodulation process needs to be performed within the FPGA. In addition, it is recommended that the IF be set at 70MHz for optimal operation. However, Figure 4.2 shows that the highest sampling rate is only 65Msps. The designer therefore has to employ the use of under-sampling to achieve the demodulation. Since this is beyond the scope of the project, and since the receiver filters on the board are also optimised for 70MHz operation (preventing off-board demodulation and conversion to base-band frequencies), the channel needs to be implemented within the software radio. This simulation process adds yet another dynamic to the usefulness of the software radio – that of high-speed channel simulation.

4.4 Conclusion

The Alcatel Altech Telecoms flexible radio platform offers many of the features and flexibility inherent in the software radio concept. Generally, the device is capable of high speed digital processing and is ideally suited to critically important system timing. It has proved a highly useful tool in developing various CDMA and digital transmission system entities. The next logical step is to examine some of the synthesis and design tools and environments employed to implement the communication system entities.

Chapter 5 Design and Synthesis

Much of the flexibility of the software radio concept is derived from its integration of software methodology and the abstraction of hardware issues from the developer. This situation requires suitable design and synthesis tools. However, as introduced in the previous chapter, the success of the integration of these two fundamentally different approaches requires the developer to be competent in both. This is a fairly broad requirement, but one that is necessary for the successful use of the relevant design and synthesis tools. This chapter will initially explore these tools, both from a programmatic and implementation aspect, and more importantly, present skills and techniques learned. These will serve to highlight the shortcomings of current design and synthesis tools, while expounding the implementation methodology as suited to the system hardware. This chapter represents the core in terms of introducing the main problem solving techniques employed in this thesis, and without this basis, no data collection would have been possible.

5.1 Design Tools

Many traditional software constructs and techniques are available in current FPGA design tools. However, when designing for an FPGA, the situation is fairly different to the traditional processor target. In the case of the FPGA, the software tools are used in the process of describing the operation of an entity on a set of inputs to produce the desired combinational or sequential outputs. This paradigm shift from a serial operation to the parallel case often leaves a traditional software programmer a little less than settled. Nevertheless, this chapter will not only present the findings of the available design tools, but also the appropriate procedure in optimising the resulting designs. The language of choice in this project has been VHDL. Specific design tools used are Synopsys FPGA Express 3.4 (for VHDL synthesis) and Max+Plus II software for compiling exported EDIF files for the target Altera devices.

5.1.1 VHDL

VHDL is a language used in the high-level description of low-level, modular electronic systems. As the Synopsys team records in the reference manual for their VHDL compiler (FPGA Express) – found in [23] – in 1980, the United States initiated a Very High Speed Integrated Circuits programme. This highlighted the need for a common means of describing two fundamental aspects of the circuitry:

- The decomposition of designs into modules and, therefore, the accurate description of the interconnection of these modules.
- The description of the functional process embodied in a module using traditional software constructs and techniques.

The result, following in 1982, was Very High Speed Integrated Circuits Hardware Description Language (VHDL), a language that has evolved over the past two decades and is now adopted as a standard by the IEEE. This language has been the pivot point in terms of developing techniques for the implementation of the CDMA system. This section will not look at the programming issues as much as the design methodology used in exploiting this programming technique for optimal results.

Describing a system in VHDL

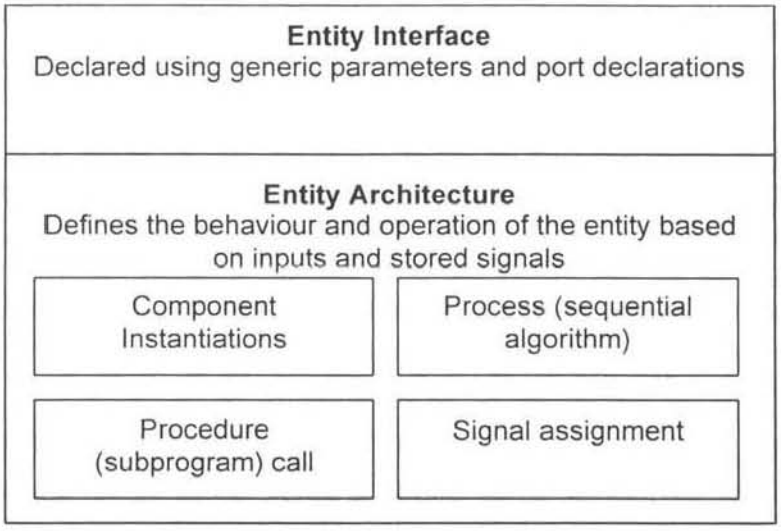

Figure 5.1 – VHDL Entity Interface and Architecture

A system is described using an entity interface along with associated entity architecture. This highly structured approach is an ideal method for the description of functional blocks within a software radio. The interface is used to define two fundamental aspects of a logic block. First, the port signals, comprising a set of input, output and bi-directional ports, are used to communicate digital information with the block. Second, generic parameters are used to quickly adapt the width and nature of these input buses, and provide algorithmic information for the architectural definition. Generic parameters create a high degree of flexibility, since the nature of the interface and underlying algorithm can be easily changed at design time without any recoding.

The operation, or architecture, of the block is closely tied to the interface. Operations on the input data are achieved using four main tools. The instantiation of other entities (in the form of components) reveals the true modular power of VHDL. These components have their ports mapped to signals or ports of the encapsulating entity and extend the functioning of the entity by reusing debugged and functional code. Processes are sets of sequential statements that appear as concurrent statements to higher levels of the architecture declaration. Common programming constructs such as loops and conditional branches are available. Sequential

statements interact with their timing signals through monitored events (such as rising or falling edges). Similar to a process, although not encapsulated in an entity, and not timed by specific signals, are subprograms. Finally, signal assignment, of both internal signals and port signals, is achieved in the architecture.

The ports and signals mentioned may take on one of many available data types. In this regard, VHDL is a strongly typed language, and casting of signal types is not at all easy. Types that are supported include Boolean, bit, integer, natural, positive, string and bit_vector. Further, these data types may be stored in vectors, and two-dimensional arrays (which are essentially arrays of arrays). The developer also has the freedom to create new data types and so may vary the bit-width of new data types to suit the application.

As can be seen, VHDL offers a plethora of means by which a system operation may be described. In addition, the nature of the construction of the language lends itself well to the development of functional blocks for software radio design.

VHDL Design Methodology

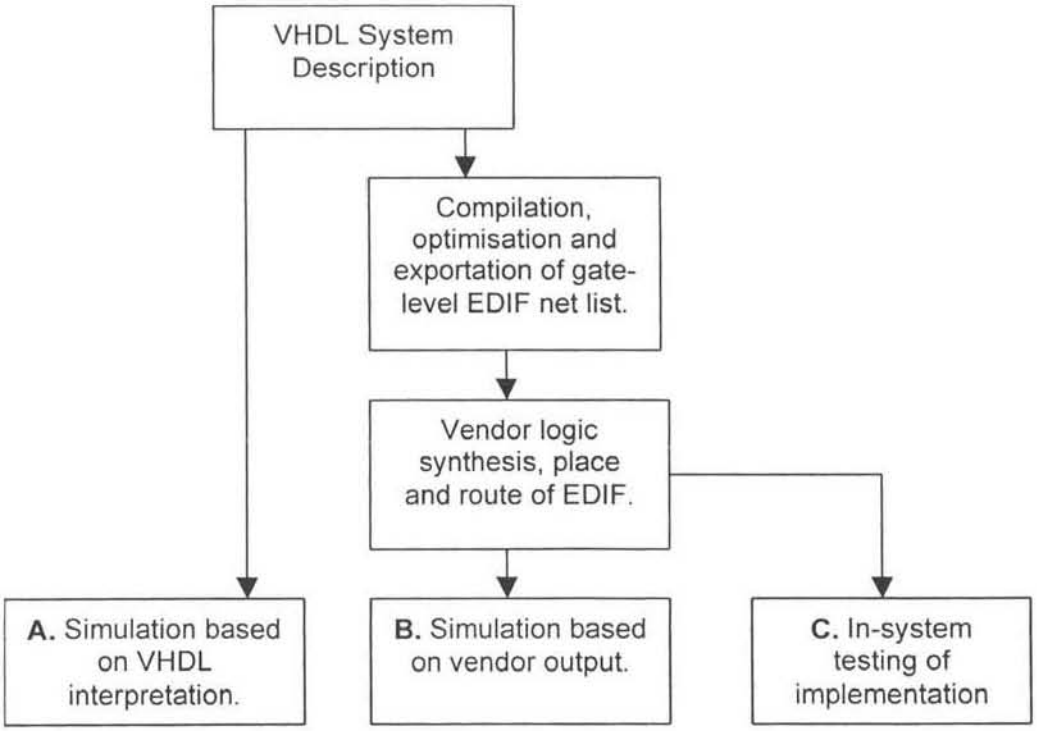

Figure 5.2 – VHDL Design Methodology

The starting point of every design is its VHDL description, generating the interfaces between entities and describing the function of each entity. It is the testing and debugging of this code that is crucial, and the process will ultimately determine the suitability of the code for synthesis purposes. Three options are then available to the developer to debug the VHDL code.

Simulation based on VHDL interpretation

VHDL simulators are suitable for removing syntactical errors, and for isolating a limited number of logical errors. However, they perform poorly as tools in terms of implementation issues such as language support, and the optimisation of speed and resource usage in FPGA's. Generally, they are designed to provide a full suite of VHDL constructs and data types. However, not all synthesis tools offer corresponding facilities. The result of this can be a well-simulated piece of code that does not compile under a synthesis environment. Further shortcomings are related to the implementation of the design. The simulator will generally "run" the VHDL code, giving little indication of how the design will perform from a timing perspective or a resource usage perspective in vendor-specific tools. Ultimately, the simulator cannot operate independently of the vendor's implementation tool, even though it may be able to use its own algorithms for gate delays and usage statistics. Actually, simulation tools may spur the designer along a highly undesirable route. Newer simulators can operate with EDIF gate-level net lists, allowing tighter integration with implementation tools.

Simulation based on vendor output

When the developer utilises the full range of development tools to arrive at a simulation, the results are more promising. The VHDL code runs through the same process that it will follow for the final implementation. This guarantees that the simulation is based on timing and resource usage derived from an actual implementation, not an algorithmic version. The result is that the simulations, while providing accurate input to output mappings (in terms of the design), will also perform comparably on the final target. However, there are several shortcomings in this approach. First, the developer has to design a suitable test-bed that will inject the correct information into the simulation environment. As the software radio concept is embraced, it is very quickly realised that this information is easily generated within the software radio environment, and often overcomes vendor-specific (and often awkward) simulation test-bed techniques. Second, the diagram of the VHDL design methodology does not accurately reflect the time spent in each stage. In fact, 90% of the implementation time is spent compiling, optimising, synthesising, routing and mapping the design. As a result, it is just as quick to set up the simulation as download the design to the final platform. In terms of usefulness within the design context, it is often far more productive to observe the system operating on the final platform. The argument for simulating the design loses its impact when, added to these arguments, is the realisation that the software radio is essentially a logic device anyway, and whether the design is tested in this or a comparable PC-simulated environment is fairly arbitrary.

In-system testing of implementation

Having the design tested on the final platform has many advantages and overcomes some shortcomings of the previous methods. Many of the advantages have been highlighted, but possibly the strongest driving factor is the issue of resource partitioning. The system may be designed to reside on a software radio where the tasks are shared amongst an array of re-configurable logic devices. To simulate the interaction of these devices is often impossible, and ignores the flexibility of the software radio platform. It was generally the experience in this project that the best design methodology was to follow the in-system testing of the implementation

5.1.2 Vendor-Specific Design Tools

While providing the necessary synthesis tools, vendors often supply an integrated environment in which design is also possible. Here, the developer is afforded the opportunity to mix various design sources in a single package. Max+Plus II was used to mix graphical design and EDIF net lists in a complementary manner.

5.2 Synthesis Tools

Synthesis tools transform VHDL code into a file that can be used for downloading to an FPGA device. In terms of this project, the synthesis operation is shared between Synopsys FPGA Express and Max+Plus II. FPGA Express is used to create an optimised net list. Once it has successfully been reduced to its net-list state, FPGA Express can analyse the timing constraints of the design. At several points in the thesis, the design results will be presented in terms of logic cell usage and speed. The speed is the estimated maximum clocking frequency as reported by FPGA Express, while the logic cell usage is a parameter reported by Max+Plus II. Max+Plus II is used to extract the netlist, synthesise the logic, partition it, fit it, extract timing information and finally assemble the project to a file for downloading. Synthesis tools are designed to hide many of the hardware issues from the programmer. Further, many classical digital design methods become redundant in an environment where logic minimisation, state machine creation and arithmetic operations are automatically implemented. Often, these implementations are more optimal than careful paper solutions of the same problem. Imperative in the design methodology is that VHDL code is constructed with the synthesis stage in mind.

5.3 Designing for Synthesis

The most important aspect of the methodology involved in creating the entities in this project has been the mastering of the ability to design for synthesis. There is a distinct difference between the code generated for simulation and that for synthesis. It is at this point that the

developer needs to be firmly grounded in the necessary programming principles while constantly being aware of the target hardware issues. This issue exists paradoxically in an environment where hardware-related issues are purposely abstracted from the developer. However, only those synthesis issues pertinent to the project will be investigated. To this end, data storage and data types need to be examined, and this can only be achieved in light of the data manipulation techniques that have been employed in the project.

5.4 Data Manipulation Techniques

The manipulation of data in a CDMA system involves a spectrum of operations ranging from those at bit-level through to fully specified mathematical procedures. The complexity of the logic required for the implementation of many mathematical functions precludes their extensive use in an FPGA where there is no dedicated adder or multiplier. However, considering the proposed system, it is obvious that functions such as addition, subtraction, multiplication and combinations thereof have to be included. For instance, the familiar multiply-and-accumulate function, native to most DSP's, has to be implemented on the FPGA. This clearly requires a more fully developed data type, and the choice thereof hinges primarily on the quest for reduced resource usage and increased speed. Mastering data manipulation requires a careful comparison of the different options. To this end, the basic evaluation is performed primarily between signed and unsigned data types, and their relevant qualities. Table 5.1 will highlight some of the necessary data manipulation techniques, while giving a good comparison in the signed and unsigned question:

Table 5.1 – VHDL Common Entity Resource Usage and Speed Analysis

Data Manipulation	Parameters	Logic Cells	Speed (MHz)
Signed Addition	Width=6	6	96
Unsigned Addition	Width=6; MSB=sign	17	66
Signed Multiplication	Width=6	78	337
Unsigned Multiplication	Width=6	54	132
Signed MAC	Width=6; Samples=6	251	23
Unsigned MAC	Width=6; Samples=6	302	14

In Table 5.1, unsigned operations have been performed preserving the sign in the most significant bit. The unsigned adder follows the following philosophy:

```
If MSB of Input2= '1'
```

```
    Result = Input1 - Input2
Else
    Result = Input1 + Input2
```

Clearly, two entities need to be instantiated for this operation, explaining the fact that this component is more than twice the size of the signed adder. For the multiplication, however, the roles reverse, and the unsigned multiplication entity is considerably smaller than its signed counterpart. However, it is also subject to a slower implementation. In the multiply and accumulate, the signed operation is clearly preferable. However, as will be discovered later in the actual implementation, the unsigned implementation can be optimised for specific conditions where it is more compact than the signed alternative. However, due to addition and subtraction bottlenecks, the unsigned version still has a slower clock response.

Many communications operation include and have to manipulate phase information. This necessitates the use of complex data manipulation. However, the IEEE has left the VHDL Complex class undefined. Table 5.2 hints at possible reasons for this.

Table 5.2 – VHDL Complex Entity Resource Usage and Speed Analysis

Complex Data Manipulation	Parameters	Logic Cells	Speed (MHz)
Multiplication	Width=6	322	337
Addition	Width=6	14	91

Table 5.2 shows that multiplying two complex numbers, whose complex components are each 6-bits wide, produces an entity with a size greater than a signed MAC entity, operating with 6 input numbers, each 6-bits wide. The resulting inefficiencies with complex number manipulation preclude their use in FPGA implementations. As a result, the communications entities need to operate under synchronous conditions. The fundamental mathematical manipulations have been investigated. There are also bit-level manipulations that need to be considered.

One of the fundamental techniques in this project is the mastering of manipulating high-speed bit streams. This includes re-ordering and extracting specific numerical values from the bit stream. To solve this particular problem, a developer with a strong processor-based programming background may be tempted to use arrays of specific-width data types and then address each of these elements individually, in order to extract a specific numerical value. However, it is often far easier to implement a large unsigned (or signed) standard logic vector. This allows for easy shifting of new data into the storage area, while simultaneously allowing bit-sized slice manipulation of the data.

Suppose a bit stream enters an entity, along with a clock and a supplied data index, and the entity is capable of storing a pre-determined number of data elements (N), each of a specific bit-width (B). The entity also has an output (of width B) that is the currently indexed data element:

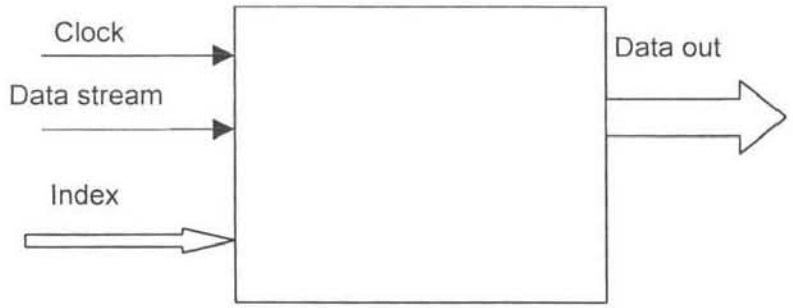

Figure 5.3 – A Simple Bit Manipulation Entity

Figure 5.3 shows a fairly simple system that acts as a storing serial to parallel converter. There are two approaches to solving this problem, both achieving the same end, but with vastly different performance.

5.4.1 Bit-Level Manipulation Approach 1

Since the data out is merely an element of a predetermined width, an initial attempt may be to instantiate an array of N elements, each with bit-width set to B. Then, as the data arrives, it can be clocked into the storage area. The algorithm is fairly simple:

```
On each clock edge
     Shift element I to the left
     Store current data bit in LSB position
     If bit counter J = B
          Set J to 0
          If I = N
               Set I to 0
          Else
               Increment I
          End
     Else
          Increment J
     End
End
```

Then, the output data is simply element(Index).

5.4.2 Bit-Level Manipulation Approach 2

The shortcoming of approach 1 is its complex addressing routine. The solution is to have a linear address space in the form of a single register. Then, as the data arrives, so one shifts the register to the left and the new data bit is stored in the LSB position. The porting of the correct data element is then seen in the following pseudo-code algorithm:

```
Store register in tmpData
Shift tmpData to right by Index*B
Output = tmpData(B-1 downto 0)
```

5.4.3 Comparison of Approach 1 and Approach 2

Both of these approaches describe the identical process. However, the currently available synthesis tools are not sufficiently advanced to reduce the two approaches to the same implementation. As a result, approach 1 can be clocked at a maximum of 38MHz (consuming 123 logic cells), while approach 2 produces an entity of only 104 logic cells, at a speed of 47MHz.

Two basic data manipulation types have been considered: mathematical and bit-level. As can be seen, the manipulation to be performed on a data element determines the data and storage types, and these will be explored further now.

5.5 Data Storage and Data Types

Due to its ubiquitous nature within the IEEE VHDL standard, the STD_LOGIC type forms the basic element of all the types that have been implemented in the project. STD_LOGIC is a sub-type of STD_ULOGIC that can take on the following values:

```
TYPE std_ulogic IS ( 'U',  -- Uninitialized
                     'X',  -- Forcing  Unknown
                     '0',  -- Forcing  0
                     '1',  -- Forcing  1
                     'Z',  -- High Impedance
                     'W',  -- Weak     Unknown
                     'L',  -- Weak     0
                     'H',  -- Weak     1
                     '-'   -- Don't care );
```

STD_LOGIC is simply a sub-type of this type where it has an associated resolution function. Here, a look-up table is employed to determine, or resolve, the value driven on a line if more than one STD_ULOGIC value attempts to drive the signal. This standard logic type forms the basis of two fundamental types:

```
type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;
type SIGNED is array (NATURAL range <>) of STD_LOGIC;
```

Collectively, these three types form the foundation of the techniques employed in data storage and type determination. Due to the high-speed implementations and low overhead of using large UNSIGNED (or SIGNED) vectors, these are preferable for synthesising big storage elements. Not only can this technique be used in bit-level manipulation, but also when wanting to store rapidly changing system values which can then be used in more complex mathematical functions. This is used in preference to array storage for the reasons previously examined in the data manipulation section.

Once the data storage technique has been specified, the choice of data type follows automatically. The reason for this lies in the strongly typed nature of VHDL. The preferred data type is either SIGNED or UNSIGNED since the underlying type in the data storage elements will then be the same. These data types are chosen in preference to integer or natural types. This is because, if data is stored in long SIGNED or UNSIGNED vectors, it cannot then be transferred to one of the more complex data types without the aid of a type conversion function. These functions, while readily available, have the following disadvantages:

- They may introduce hidden overhead
- They reduce the readability of the code
- They may have undesirable side-effects

The preferred data type is then logically either SIGNED or UNSIGNED, as opposed to integers or naturals.

5.6 Conclusion

Designing in VHDL involves a process of describing the desired operation of a complex digital circuit. This description takes place in a software language, and out of this, the software radio derives its greatest strength and flexibility. This approach also helps immensely in reducing the time required to construct complex circuits. However, the designer has to design within the context of the currently available synthesis tools. As a result, the degree to which the programmer can overcome the abstraction of the hardware issues is, in many cases, the degree to which the design is a success. Also, synthesis tools evolve and, as a result, more complex algorithms are developed. The impact of this will be that any number of components, each with process descriptions aimed at the same result, will produce identical implementations, in spite of slight underlying description differences. At the moment, however, the developer is influenced by the choice of synthesis tool, and has to design with this in mind.

Chapter 6 System Overview

Until this point, this thesis has presented the reader with a fairly comprehensive background. This has included both a theoretical study of spread spectrum communication techniques, balanced with a look at some of the practical issues relating to system implementation. With this background work, it is now possible to propose a practical communications system. This chapter will give a brief overview of the system functional blocks employed in such a system, while the subsequent chapters will delve into the intricacies of several of them. The blocks have been implemented on Alcatel Altech Telecomm's flexible radio platform (introduced in section 4.3).

6.1 The System Block Diagram

This diagram is presented for the downlink of a practical CDMA system employing digital transmitter and receiver filters, while employing Walsh codes and a suitable MMSE structure to resolve the channel information.

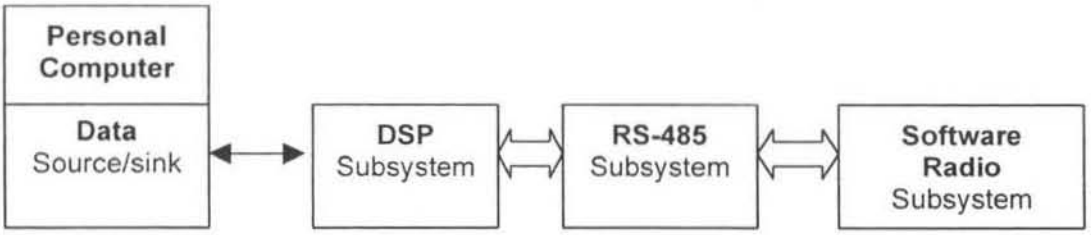

Figure 6.1 – The Project Block Diagram

Figure 6.1 shows the essential components of the overall system. This design has been adopted, since the Alcatel Altech Telecomms engineering team instigated it. The Personal Computer acts as a data source/sink, giving the designer control over the nature of the data sent to the physical layer. The DSP is primarily responsible for packing the data correctly for the software radio. This includes framing and packing the data into bytes. A daughter card is responsible for converting the high-speed serial signals from the DSP subsystem to RS-485. The software radio subsystem includes a corresponding RS-485 communications module. These two RS-485 conversion modules introduce a conversion delay that has fairly severe implications. Chapter 7 will deal further with the issues of interfacing the software radio subsystem and the DSP subsystem. However, the core of the implementation process deals with the elements within the software radio. Here, the CDMA physical layer is implemented.

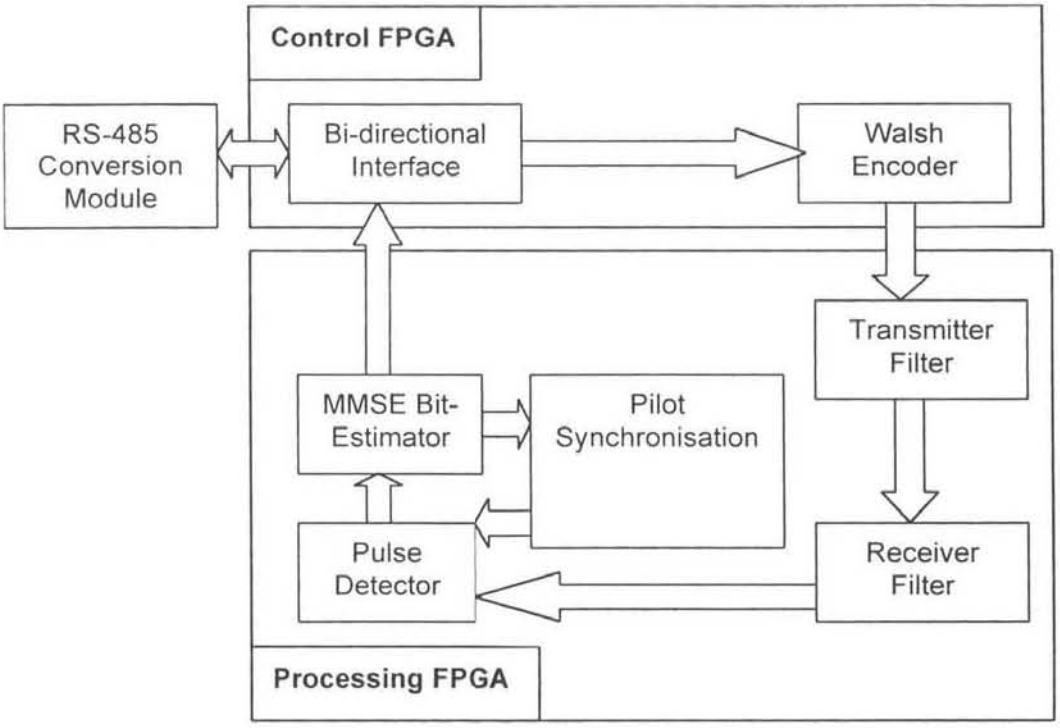

Figure 6.2 – Detailed System Block Diagram

Figure 6.2 outlines the main elements residing in the software radio. The bi-directional interface, connected to the RS-485 conversion module, is responsible for generating a variety of clock signals for the communications bus that exists between the software radio and the DSP subsystem. These signals allow for proper framing and timing within the DSP subsystem, and will be explained in great detail in Chapter 7 . This interface also serves to buffer and correctly pack the data from the MMSE bit estimator and send it on to the DSP. The bi-directional interface is also responsible for unpacking the information received from the DSP subsystem and presenting it to the Walsh encoder in a suitable format. This allows the Walsh encoder to chip the data for several users simultaneously and then superpose it. Chapter 8 will deal with the implementation of the Walsh encoder, introducing the idea of the code database as a "storage and chipping" technique. Further, there are two filters included in the processing FPGA. 4.3, The Alcatel Altech Telecoms Flexible Radio Platform dealt with the support hardware that is included in the software radio used in this project. Unfortunately, while transmit mixers are included there is no comparable hardware to return a received signal to its base-band form. In fact, the original intent was for designers to incorporate a receiver mixer in the software radio. Since this is outside the scope of this thesis, the transmitted signals are simply tied internally to the receiver filter. In Chapter 11 , this transmitted signal will be distorted with internally generated AWGN and Rayleigh fading before being joined to the receiver filter. Chapter 9 will examine both the transmitter and receiver filter in detail, showing how the software radio is useful for moving such filtering operations into the digital domain. Finite Impulse Response digital filtering techniques are employed to limit the signal

bandwidth, while suppressing inter-symbol interference and maximising the signal-to-noise ratio at the input to the detector. The pulse detector entity is responsible for sampling the stream received from the receiver filter. This entity has its sample point controlled through a feedback decision made by the pilot synchroniser. The pulse detector is capable of shifting in time to present a different set of sampled chips to the MMSE bit estimator. This estimator is responsible for accumulating these chips and processing them to produce the bit estimate. Chapter 10 extends some of the theoretical information on multi-user detectors to the point of implementation, examining the limitations of an FPGA-only solution to the problem of detection. Included is a simple system for synchronising to a pilot signal, along with a performance analysis. Finally, in Chapter 11 , the system is subject to testing in a simulated channel environment. Here, additive white Gaussian noise and Rayleigh fading implementations are examined.

From this point onward, the reader will be exposed to more of the practical implementation issues associated with this type of CMDA system. Finally, the measured performance of the system will be presented in the form of a conclusion.

Chapter 7 Interfacing

This chapter will examine the issue of interfacing the software radio to the DSP subsystem. This will include a close look at the timing specifications of the communications bus. As the system is developed, timing complexities will be highlighted, and solutions proposed. Furthermore, the mechanism for generating a serial stream of successive user's bits from an input stream of successive user's bytes will be broached. This last point will allow for a brief investigation into the timing and resource restrictions in designing such an interface in VHDL. This will include a discussion on a few buffering techniques.

7.1 Interfacing Overview

The software radio communicates with the PC using an RS-485 connection as the physical layer that transports SCSA telecommunications protocol data. Thus, the TTL logic levels, generated on the DSP card by an SC4000 device (to implement an SC-BUS), are first converted to paired RS-485 differential signals before being transmitted to the software radio. This conversion is achieved on a daughter-board that plugs into the PC. On the software radio, the signals are converted back to digital logic levels before being routed to specific pins of both the FPGA's in the software radio. Clearly, on a feed-through net, the signal is delayed by four physical layer signal-type conversions. This induces a round-route delay of 200ns, an important factor for the timing of the system. One last important note, before the timing and protocol are detailed, is the fact that the software radio may be used as the clock source for the timing of the bus.

7.1.1 The SC-BUS Standard

There are three different bit-timing configurations

Table 7.1 – SC-BUS Standard Communications Parameters

Bit Rate (Mbps)	Number of Timeslots
2.048	32
4.096	64
8.192	128

Each timeslot consists of 8 bits. From Table 7.1, it is possible to calculate the frame rate (8kHz), which is independent of the bit rate. The system that was implemented, used a 2.048MHz bit rate. Further, there are 16 channels ($SD_{0..16}$), each operating at 2.048Mbps. These have been divided into 8 transmit and 8 receive channels. The transmit channels (with respect to the DSP card) take on channels SD_i where $i \in \{1,3,5..15\}$ and the receive channels take on channels SD_i where $i \in \{0,2,4..14\}$. Other fundamental signals include FSYNC (frame

sync), SCLK (serial data clock) and SCLKx2 (double speed serial data clock – imperative for bit sampling.

SC-BUS Timing

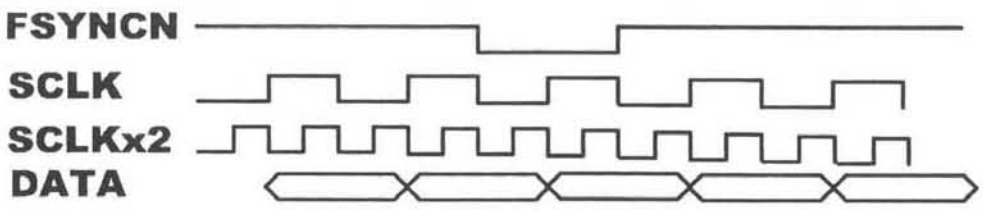

Figure 7.1 – The SCBUS Timing Diagram

Figure 7.1 shows the synchronous relationship between the serial data clocks, the frame synch pulse and the data bits. While the DSP system acts as the data source, from a design perspective, it is useful to allow the software radio to act as the master clock source. However, this presents a critical timing problem. Since the single-rate serial data clock is also sent over the SC-BUS and is also subject to delays, the associated data bit will arrive at least 200ns after SCLK is driven high. At 2.048Mbps, the bit duration is 488ns, thus, to sample in the middle of the bit (244ns) is dangerously close to the 200ns boundary. In practice, the data source (the DSP card) is interrupt driven. Thus, when it receives a bit boundary pulse, it has to generate the data from transmission. The return rate is highly dependent on the complexity of the transmitter interrupt handler. On testing, it was seen that the return delay often violated the 244ns critical delay. As a result, 50% sampling had to be avoided (i.e. Sampling in the middle of the returned bit). This is where the double rate serial clock is extremely useful, since it allows for 75% sampling (i.e. sampling at ¾ of the duration of the bit), which adequately allows for the bit-level transmission delays.

Implementation in VHDL

At this point, to demonstrate the implementation of this timing philosophy, it will be useful to examine a VHDL code fragment.

```
1.   if (SCLKx2'event and SCLKx2 = '1') then
2.   if ((FSYNCN = '0') and (BitCounter /= 511)) then
3.        tmpBitCounter      := 511;
4.        FrameByteCounter   <= "111";
5.   elseif (BitCounter = 511) then
6.        tmpBitCounter := 0;
7.   else
8.        tmpBitCounter := BitCounter + 1;
```

```
9.   end if
10.       if (tmpBitCounter mod 2 = 0) then
11.            StoreBuffer <= StoreBuffer SRL 1;
12.       elsif (tmpBitCounter mod 2 = 1) then
13.            StoreBuffer(255) <= MAC2PHY;
14.       end if;
15.       BitCounter <= tmpBitCounter;
16.  end if;
```

This code fragment overcomes the 50% sampling error. As seen in line 1, the double-rate serial clock clocks the entity, resulting in counting process seen in Figure 7.2.

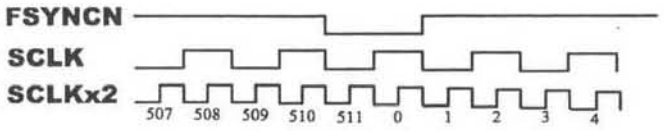

Figure 7.2 – VHDL Counting and the SC-BUS Signals

The next step in the entity is to store the incoming data bit in a reserved buffer. On every even clock edge, the buffer is shifted right one, while every odd clock edge, the received bit is stored at the beginning of the buffer. In this manner, the received bits are sampled at 75% of the bit duration, overcoming otherwise serious timing discrepancies.

7.2 Processing the DSP to Software Radio Stream

The data sent from the DSP card is formatted as a series of user bytes, each one stored in a time slot. Since there are 32 time slots, and only 16 users are supported by the system presently, the second half of the frame will be discarded. For the data bits to be chipped and sent appropriately, the stream needs to contain successive user bits, not bytes. Primarily, this is because the DSP system is not sufficiently powerful to format the data stream so as to have successive user's bits making it up. The solution is to create a buffering entity that is also capable of re-ordering the received bits. Conversely, the process needs to be reversed when sending data back to the DSP system. Here, the successive user's bits need to be constructed into successive user's bytes, allowing for the correct reception on the DSP board.

7.2.1 The Reordering Process

The reordering of the stream occurs in a dual array of bits. These two buffers, a storage buffer and a shipping buffer, are declared as long unsigned standard logic arrays. This allows for the implementation of rapid shift registers. The previous code fragment shows how the storage buffer is generated.

```
1.   if (SCLKx2'event and SCLKx2 = '0' ) then
2.        if (BitCounter = 511) then --we need to reset the counters
3.        ShipBuffer         <= StoreBuffer;
4.        tmpByteCounter     := 1;
5.        PHYDataOut         <= StoreBuffer(0);
6.   else
7.        tmpByteCounter     := ByteCounter;
8.   end if;
9.   if ((BitCounter mod 2 = 1) and (BitCounter /= 511)) then
10.            PHYDataOut <= ShipBuffer(tmpByteCounter * 8);
11.            tmpByteCounter := tmpByteCounter + 1;
12.            if (tmpByteCounter = 32) then
13.                 ShipBuffer <= ShipBuffer SRL 1;
14.                 tmpByteCounter := 0;
15.            end if;
16.       end if;
17.       ByteCounter <= tmpByteCounter;
18.  end if;
```

This code fragment has exactly the same value for the BitCounter variable as in the previous fragment. However, to achieve the high level of synchronization, this process is clocked on the falling edge of the double-rate serial clock. Thus, as soon as the bit counter reaches a value of 511, and the storage buffer has been completely filled, it is copied to the shipping buffer (line 2-3). At the same time, the first bit from the storage buffer is placed on the data out (line 5). Note that with VHDL, the signal assignment is only valid once the process is complete, and this is why the storage buffer is used to source this data bit, and not the shipping buffer (line 5). Note also that, from the timing analysis, every odd falling edge of the double-rate serial clock corresponds to the beginning of a bit period. Thus, line 9 checks for this condition, while avoiding the first time this condition is true (when the bit counter is 511). Lines 10 to 15 achieve the re-ordering process. By referencing 8 times the current byte counter value in the shipping buffer, the next bit to be transmitted can be sent. Once all 32 time slots have been processed in this manner, the shipping buffer is shifted to the right (line 13), and byte counter is reset. This will occur for each of the 8 bits per time slot. In this manner, the input stream is mapped to the output stream:

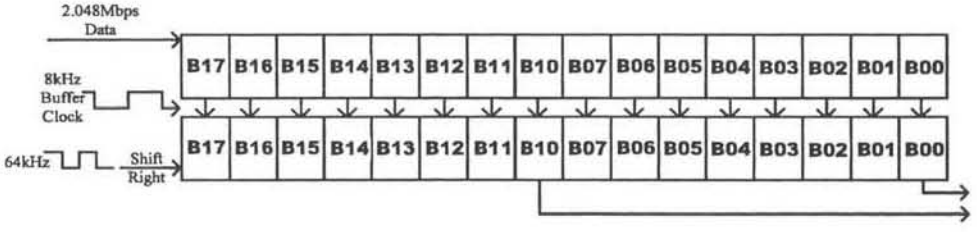

Figure 7.3 – Bit-Level Transpose on Data Stream

In Figure 7.3, only two byte are shown, where Bxy, is bit y of byte x. In this manner, the bits are shipped out of the entity first in byte order (user 0 is sent first), and then in bit order (bit 0 of user 0 is sent first).

The Inverse Ordering Process

Before the user bits may be sent to the DSP, they need to be re-packed in user bytes. An entity not unlike the above-mentioned is employed for this task. However, the process is reversed so as to present the DSP with intelligible data.

7.2.2 Serial to Parallel Conversion

A further facet of the interfacing procedure is to create an entity capable of extracting bytes and words from the serial data stream. This entity finds its use when dealing with a parameterised system. For instance, if the system is set up to generate AWGN of a specific SNR, then the parameter would be the desired level of the noise. Another example is a 16-bit number where the individual bits are code-usage flags. In other words, if the bit is set, then the corresponding chipping code in the chipping database must be enabled. These parameters may be conveniently encoded as bytes and words by the DSP and then transmitted serially to the software radio. To carry out the addition or multiplication with these parameters as inputs, the stream needs to undergo a serial to parallel conversion. Furthermore, the parallel data needs to be synchronous with the allocated SC-BUS time slots.

```
1.   DoPacking : process(SCLK, FSYNCN)
2.        variable tmpLocalByte : IEEE.numeric_std.UNSIGNED(7 downto
  0);
3.   begin
4.        if (SCLK'event and SCLK = '0') then
5.             if (FSYNCN = '0') OR (ByteCounter = "111") then
6.                  ByteCounter <= "000";
7.             elsif (ByteCounter < "111") then
8.                  ByteCounter <= ByteCounter + "001";
9.             end if;
10.            tmpLocalByte    := LocalByte SLL 1;
11.            tmpLocalByte(0) := BitStream;
12.            LocalByte       <= tmpLocalByte;
13.       end if;
14.  end process;
15.
16.  DoShipping : process(SCLK)
17.  begin
18.       if (SCLK'event and SCLK = '1') then
19.            if (ByteCounter = "111") then
```

```
20.                 ByteCLK <= '1';
21.                 ByteOut <= LocalByte;
22.             elsif (ByteCounter = "011") then
23.                 ByteCLK <= '0';
24.             end if;
25.         end if;
26.  end process;
```

The entity responsible for packing the bytes is broken into two logical processes. This first accumulates the byte information, while the second, responding to the complementary clock edge, makes the constructed byte, along with a positive logic clock, available to other entities. Notice how line 5 once again achieves synchronism by resetting the main counter while the frame sync is low. Notice how, since the single rate serial clock is being used, the frame sync is low only once during a clock cycle, removing the need to avoid a second frame sync reset. Once the bits have been shifted in, the rising clock edge is used to map the accumulated byte to the output port of the entity.

7.3 Conclusion

This chapter has examined the issue of interfacing the software radio to the DSP subsystem. Included in this process was an examination of the timing specifications of the communications bus. With regard to this, it was seen that VHDL is a highly descriptive means of overcoming some of the timing issues, while providing high speed buffering and bit-processing entities. Once the software radio has the correct representation of the data, it needs to be spread through a high-speed chipping process. This subject is broached in the next chapter.

Chapter 8 Walsh Encoding

Fundamental in any CDMA system is an effective means of chipping the base-band data, thus spreading the information across a higher bandwidth. Previous chapters have tackled the issue of choosing the correct type of code. Now, it is necessary to investigate means of practically implementing a chipping process. Further, this particular implementation is intended to emulate a downlink, and will therefore have to chip and superpose the information from many users. This necessitates careful attention to timing issues, especially when considering the link between the spread data source and the base-band filter. This is to prevent any phase errors from being introduced in the transmitter filter. This chapter will then look at the particular approach used in implementing a Walsh encoder for 16 simultaneous users, each with a base-band data rate of 64kBps. To this end, an examination of some VHDL code fragments will also familiarise the reader with some of the more difficult implementation and timing issues.

8.1 The Entity Structure

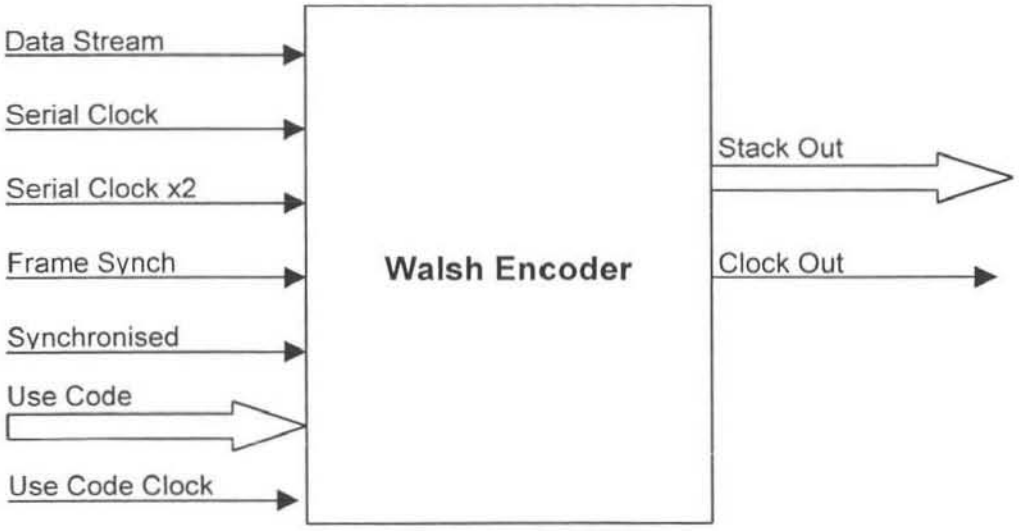

Figure 8.1 – Entity Diagram for the Walsh Encoder

The Walsh Encoder makes use of several clocks and synchronisation signals to achieve the correct chipping process. Chapter 7 introduced the reader to the interfacing concepts, along with the SC-BUS timing signals. The same serial and double-rate serial clocks are used to clock the data stream into the entity. Further, there is an entity reset pin in the form of the synchronised signal which, when low, indicates that the entity needs to enter a reset state. A further feature of the entity is the ability to isolate specific codes for chipping. This is achieved with a 15-bit wide "use code" signal that will include or exclude the corresponding code if the signal bit is set or cleared respectively. The system is capable of chipping 16 streams of user data, but the 16^{th} code is reserved for the pilot channel. As a result, only 15 bits are necessary

to completely determine the inclusion or exclusion of a user in the transmission process. The nature of the pilot signal and code will be dealt with in 10.4, Choice of Pilot Signal Code and Bit Sequence. The "use code" value is latched into the entity when the "use code clock" experiences a rising edge.

The entity has the task of generating 16 chips for every user bit. Further, there are 16 users supplying data to this entity. As a result, each chip needs to be superposed with at most 15 others to generate the entity output. This relates to 33 possible output levels per bit, and these are reflected in the "stack out" output, accompanied by its relevant clock. This clock is used to accurately provide the transmitter filter with the necessary timing for its filtering operation.

8.2 The Code Database

Central to the Walsh encoder is the storage area used for the chipping sequences. This code database is fairly dynamic in that its contents may be programmed from the DSP if necessary. This technique has been used in preference to the standard generation techniques since it affords the developer a high degree of flexibility when testing various codes. It does not rely on maximal length shift registers, or combinations thereof, and this architecture-independent approach to the code generation allows for easy code modification. However, it does present the problem of greater resource usage. The codes have to be stored in re-configurable logic, and in a manner that allows them to be easily accessed and used to spread data. The concept of the code database is simple, and allows the developer to store the codes in this manner. In terms of VHDL, the code database is declared as a long unsigned vector containing 256 (16 x 16) elements. By default, these are loaded with Walsh sequences where

$$C_i = [(16-i)\cdot 16]-1..[(15-i)\cdot 16] \quad i \in \{0..15\} \qquad \text{.....Equation 8.1}$$

Where C_i is the code associated with user i.

8.3 The Stacking Process

The chipping process ties in closely with the fact that the output is a multi-level signal. While each code is stored as a mono-polar signal (0,1), it is treated as a non-return to zero stream. The same is true for the data stream which, when multiplied with the relevant code, will also produce a NRZ stream. Essentially, this multiplication process sees the data bit either preserving the code (if it has a value of 1) or inverting it (if it has a value of –1). It is intuitive then that the maximum attainable value will be 16 (if all corresponding chips for all users are set to 1) and the minimum value will be –16 (if all corresponding chips for all users are set to –1). This translates to 33 different levels, and can only be successfully stored in a 6-bit signed

number. This then is the stack output that is fed to the transmitter filter. The following code fragment reveals the VHDL procedure:

```
1.   if (UseWalsh = '1') then
2.        tmpCurrentUserBit := PHYData;
3.        for i in 0 to 15 loop
4.             if (tmpCurrentUserBit = PNDatabase(255-i)) then
5.                  tmpPNStackIn(i) := tmpPNStackIn(i) + "000001";
6.             else
7.                  tmpPNStackIn(i) := tmpPNStackIn(i) - "000001";
8.             end if;
9.        end loop;
10.  end if;
11.  PNDatabase <= PNDatabase ROL 16;
```

Lines 1 to 10 represent the creation of the stack for a single user's current bit. Once this is complete, the database if rotated 16 bits to the left, loading bits 255 down to 240 with the appropriate code for the next user. Line 1 allows the entity to check whether or not the current Walsh code should be included in the chipping procedure. Line 4 does an effective exclusive or between the user's data bit and the relevant database bit. If they are both the same, then the output chip is 1 (line 5), otherwise it is –1 (line 7).

8.4 Conclusion

While the technique proposed in this chapter does require extended resources, it provides a flexible means of modifying the chipping sequences employed in the CDMA system. These can be altered either at design time or run time. In addition, this entity accurately mimics a base station operation in that it can send the data for up to 16 simultaneous users, each with its own chipping sequence. Once the data has been spread, it is essential to process it for transmission. This is achieved though base-band filtering in the digital domain, and is the subject of the next chapter.

Chapter 9 Base-Band Filtering

Before the spread data may be up-converted to IF, it needs to be processed to optimise a very specific set of transmission parameters. Essentially, this processing reduces to a matched filter, with a root-raised cosine transfer function. This chapter will examine both the theory and the practical implementation of base-band pulse shaping on an FPGA device. This will involve first establishing the necessity of pulse shaping and, second, exploring methods for implementing a multi-level FIR filter for the transmitter and receiver stages. To make the implementation analysis complete, there will be an investigation of several realisation structures, all aimed at optimising speed and resource usage. To conclude this section on implementation possibilities, there will be an examination of the distributed arithmetic structure, which is a fairly new and exciting technique for implementing FIR filters in FPGA devices.

9.1 Bandwidth

The communications channel under consideration is bandwidth-constrained in that it will distort a signal of wide bandwidth. [16] considers the standard representation of a digital signal as rectangular pulses:

$$p(t) = \begin{matrix} 1, & |t| \le \frac{\tau}{2} \\ 0, & otherwise \end{matrix} \qquad \text{.....Equation 9.1}$$

This has the familiar Fourier representation:

$$P(\omega) = \tau \frac{\sin(\omega\tau/2)}{\omega\tau/2} \qquad \text{.....Equation 9.2}$$

- ω is the angular frequency
- τ is the period of the rectangular pulse

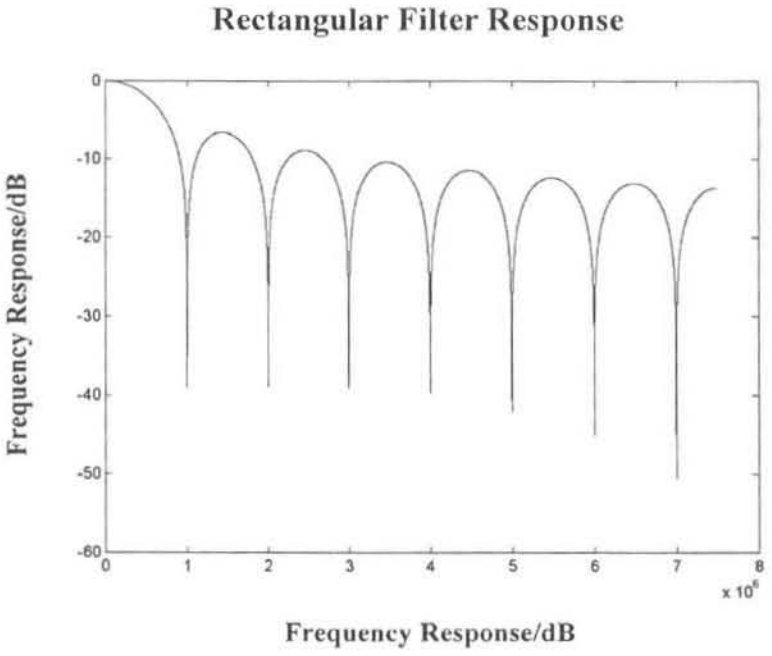

Figure 9.1 – Predicted Frequency Response of the Rectangular Filter

As seen in the Figure 9.1, the rectangular pulse has harmonics that decay very slowly. This translates to a large amount of power existing outside of the fundamental frequency of the pulses. Thus, a bandwidth limited communications channel will distort these higher frequency components. The obvious solution is to filter out the higher components by performing some time-domain shaping of the transmitted pulses. To begin a search for a suitable filter, the specifications need to be clearly delineated:

Time-Domain Requirements

- The pulse shape must have a value of zero at all pulse sampling times other than its own
- The pulse shape must have tails that decay rapidly so as to minimise the effect of timing jitter in either the pulse generation or sampling.

Frequency-Domain Requirements

- The Fourier transform of the pulse shape must approach zero for all angular frequencies greater than some pre-determined value

9.2 The Rectangular Filter

Considering the rectangular pulse, only the frequency-domain requirement is violated. A suitable response to this shortcoming would be to limit the Fourier transform where:

$$P(\omega) = \begin{matrix} \tau & \omega \le \frac{\pi}{\tau} \\ 0 & otherwise \end{matrix}$$ ..... Equation 9.3

this corresponds to the familiar time domain representation of

$$p(t) = \frac{\sin(\pi \cdot t/\tau)}{\pi \cdot t/\tau}$$ ..... Equation 9.4

Although not immediately evident from the equation, the time domain plot reveals that the tails of the pulse decay too slowly due to their dependence 1/t.

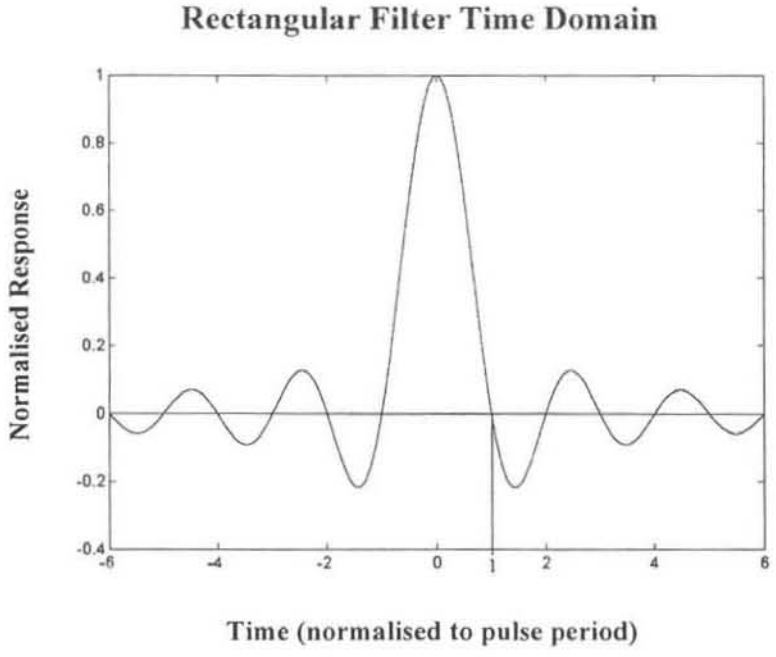

Figure 9.2 – Time Domain Response of the Rectangular Filter

There is a manner in which this response may be modified to retain its bandwidth properties while increasing the damping in the time domain.

9.3 The Raised Cosine Pulse Filter

One pulse that satisfies the two time-domain criterion and the frequency-domain criterion, is the raised cosine pulse:

$$P(\omega) = \begin{cases} \tau & 0 \le \omega \le \dfrac{\pi \cdot (1-\alpha)}{\tau} \\ \dfrac{\tau \cdot [1 + \cos\{\tau \cdot [\omega - \pi(1-\alpha)/\tau\,]/2\alpha\}]}{2} & \dfrac{\pi \cdot (1-\alpha)}{\tau} \le \omega \le \dfrac{\pi \cdot (1+\alpha)}{\tau} \\ 0 & \omega \ge \dfrac{\pi \cdot (1+\alpha)}{\tau} \end{cases}$$

Equation 9.5

This corresponds to a time domain representation of:

$$p(t) = \frac{\sin(\pi \cdot t/\tau)}{\pi \cdot t/\tau} \cdot \frac{\cos(\alpha \cdot \pi \cdot t/\tau)}{1 - (4\alpha^2 t^2)/\tau^2}$$

Equation 9.6

Before the above may be implemented, the effect of alpha on the signal needs to be investigated. For its effect on the damping of the pulse tails, this is best seen in a graphical time-domain analysis:

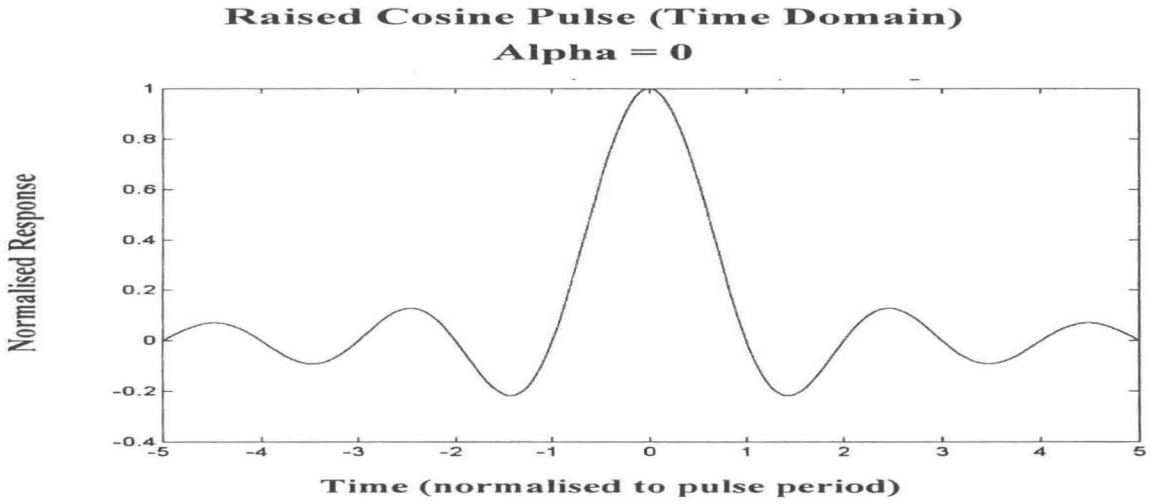

Figure 9.3 – Time Domain Response of the Raised Cosine Filter

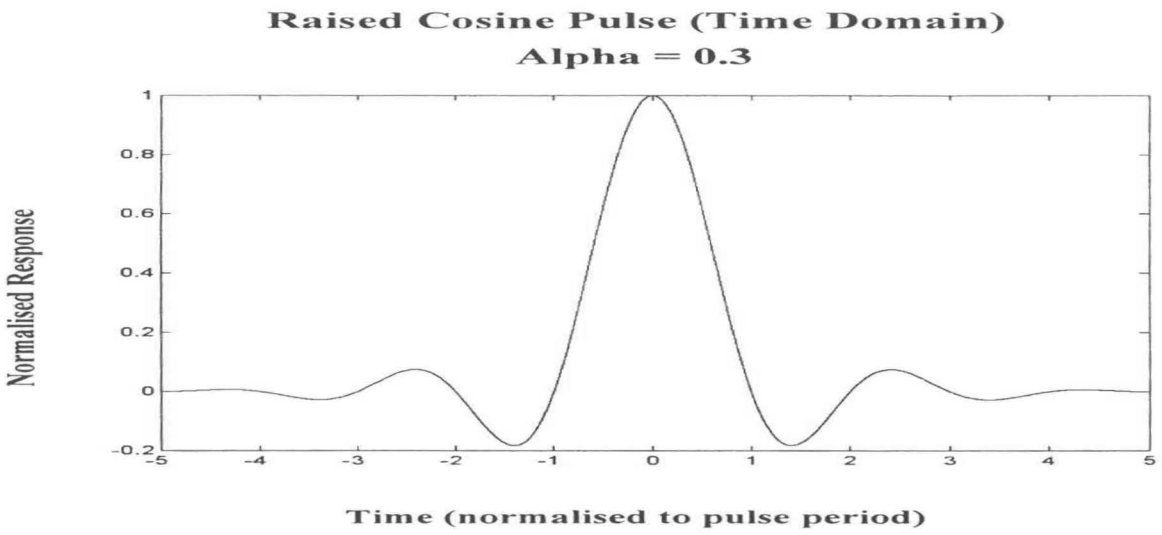

Figure 9.4 – Time Domain Response of the Raised Cosine Filter, Alpha = 0.3

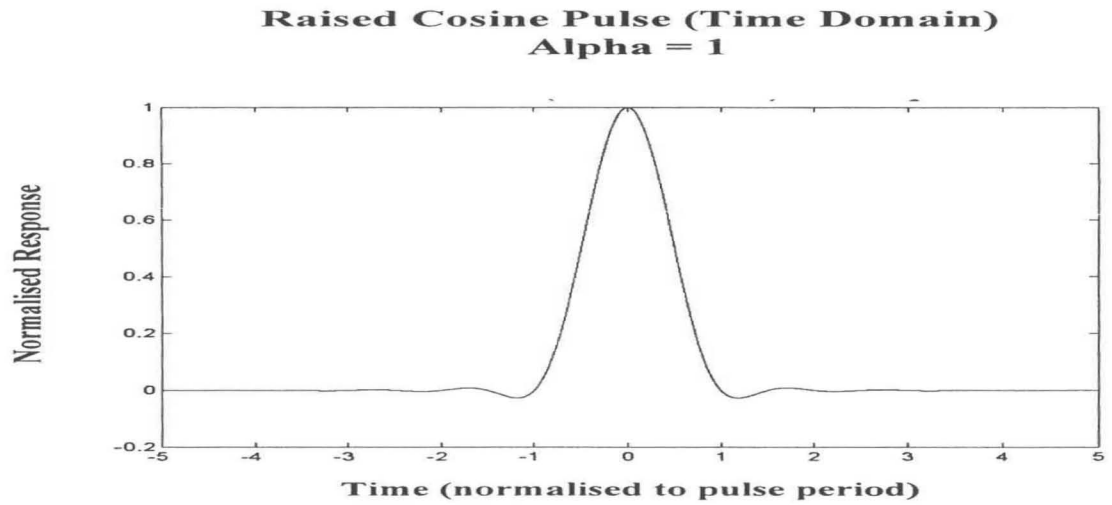

Figure 9.5 – Time Domain Response of the Raised Cosine Filter, Alpha = 1

Although the time domain benefits of raised cosine pulse shaping are immediately obvious, it is necessary to modify alpha so as to limit the required bandwidth. Examining the Fourier transform shows that the bandwidth is highly dependant on alpha – it will double as alpha is increased from 0 to 1. Feasible data transmission systems usually allow alpha to be set as low as 0.3.

9.4 Matched Filtering

Since the shaped data will be transmitted over a real channel, it will undergo some form of distortion. This distortion may include additive white Gaussian noise or Rayleigh fading. Once the data has been received, it is necessary to pass it through some form of a linear filter so as to remove as much of the channel distortion as possible.

$$r(t) = s(t) + n(t)$$ **.....Equation 9.7**

- r(t) is the received signal
- s(t) is the transmitted signal
- n(t) is the assumed AWGN, superposed on the statistically independent signal

This received signal is filtered by a linear filter with impulse response g(t) and Fourier transform G(f). The characteristics of the filter need to be designed so as to maximise the signal to noise ratio (SNR) at the input of the symbol detector.

9.4.1 Characterising the Matched Filter

The task at hand is to describe the receiver filter so as to maximise the SNR. [A.P. Clark, 1983], [17], presents a rigorous treatment of the receiver filter characteristics in the presence of noise, and relates these to the received signal. The conclusion of this analysis presents a direct relationship between the optimal receiver filter and the transmitted signal s(t).

$$g(t) = c.s(T - t)$$ **.....Equation 9.8**

This final point is the crux of the filter design, showing that, if the signal characteristics are known, then it is possible to design a filter such that the signal to noise ratio is maximised. However, this requires some prior knowledge of the parameters of the transmitted signal. This is best achieved if the signal is processed before it is transmitted. In this manner, the receiver can estimate quite accurately the nature of the received signal. From this perspective, it is imperative to design the receiver filter in conjunction with the transmitter filter.

9.5 *Designing the Transmitter and Receiver Filters*

The preceding discussion has presented a new design problem, that of designing the transmitter filter in such a manner that the receiver filter is aware of the received signal characteristics. Given that the proposed communications path is fairly simple and therefore will be subjected only to AWGN, it is now possible to propose a functional block diagram and design the frequency response of the various elements.

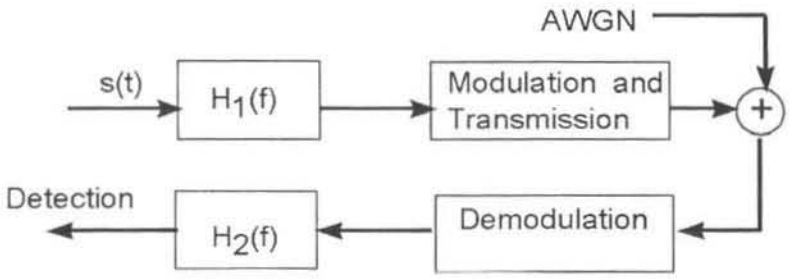

Figure 9.6 – The Matched Filter Communication Path

In Figure 9.6, the task is to design the transmitter filter function $H_1(f)$ and the receiver filter function $H_2(f)$. These need to be designed together so as to shape s(t), making use of the transmitter filter, in order that, once processed by the receiver filter, the detector will detect the received symbol with the lowest error probability. Once again, [A.P. Clark, 1983] presents a mathematical discourse unveiling the frequency response of the circuit elements. If concludes with a simple relationship between the transmitter and receiver filter transfer functions.

$$|H_2(f)| = c|H_1(f)| \qquad \text{.....Equation 9.9}$$

It is highly convenient to set c (in Equation 9.9) to 1. Under this condition, both the transmitter and receiver filter will have a response equal to the square root of the desired overall base-band response. This theory can now be applied to the implementation phase.

9.6 *Implementing the Transmitter and Receiver Filters*

The matched filter system will ultimately be implemented in high-speed digital logic, residing in an FPGA. Various aspects of the filter need to be considered in this light, since it will have a digital implementation. First, it is necessary to reconsider the communications parameters as proposed in the project specification:

- Transmission rates will be 64kBps per channel

- Each channel will be chipped with a Walsh sequence of length 16 => chipping rate of 1.024 Mcps
- A system clock speed of 132MHz

The objective now is to create a transmitter filter that is band-limited to 1.024 MHz with a matched receiver filter. For the purposes of this design, a Finite Impulse Response filter will be used.

9.6.1 The Digital FIR Filter

The FIR filter has several characteristics that make it ideal for use in the proposed communication system. Many of these advantageous traits are recorded by [Ifeachor & Jervis, , 1993], [18], and these include the following:

- FIR filters introduce no phase distortion to the communication signal. This is a requirement when considering data transmission systems
- The alternate to FIR filters is IIR (Infinite Impulse Response) filters that are realised in a recursive structure. This may introduce instability and cause the filter to malfunction.
- FIR filters are less sensitive to round-off and quantisation errors than are IIR filters.

The general formula for the FIR convolution is the following Multiply and Accumulate (MAC):

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \qquad \text{.....Equation 9.10}$$

- h(k) are the impulse response coefficients
- N is the number of filter coefficients (taps)
- x(n-k) are the delayed samples of the input waveform
- y(n) is the output (filtered) waveform based on the previous N input samples

As can be seen from the defining equation, there are N multiplications and N-1 additions. Since the FPGA is a device without a dedicated multiplier, this task needs to be carefully optimised so as to ensure not only minimal resource usage, but also maximum performance. The first step in optimising such an implementation is to investigate possible structural implementations of the filter. First, however, it is useful to design the actual coefficients that will be used in the system filters.

FIR Coefficient Design

There are two possible routes to take in designing the coefficients for the system – either by using specific design software, or by using a mathematical package such as MATLAB.

However, it is important in the design of the coefficients to ensure that the full dynamic range of the Analogue to Digital converters is used. This will ensure that the Signal to Noise Ratio is maximised from the transmitter stage, and is preserved through to the receiver stage. To achieve this, it is best to use standard filter software to calculate the floating-point coefficients. It is also possible to use MATLAB to calculate these values. However, important in the process is that these coefficients are correctly scaled. In the following simple procedure, the filter coefficients can be scaled to avoid overflows and underflows while maximising the use of the ADC range:

- Ascertain the ADC full desired range (unsigned) and record this as RANGE
- Pass the full range of input values through the filter with the floating point coefficients
- Record the maximum filter output (MAX) and the minimum filter output (MIN)
- The scaling factor is then RANGE/(MAX-MIN)

The scaling factor is then used to multiply the coefficients, which are subsequently rounded toward zero to prevent under- and over-flows, while allowing for binary representation.

9.6.2 Implementation of the Transmitter Filter

Before being used as a modulating signal in the Intermediate Frequency (IF) stage, the data awaiting transmission needs to be shaped. In terms of the matched filtering process, the transmitter filter will perform one half of the overall raised cosine filtering in the form of a root-raised cosine filter. The realisation structure for the FIR transmitter filter may be optimised to take advantage of the nature of the input data. First, the source and outputs need to be clearly defined

- Source data is stacked Walsh chips at a rate of 1.024 Mcps
- Output is 12-bit encoded pulses constructed from 8 samples, constituting one symbol at a sample rate of 8.192 Msps

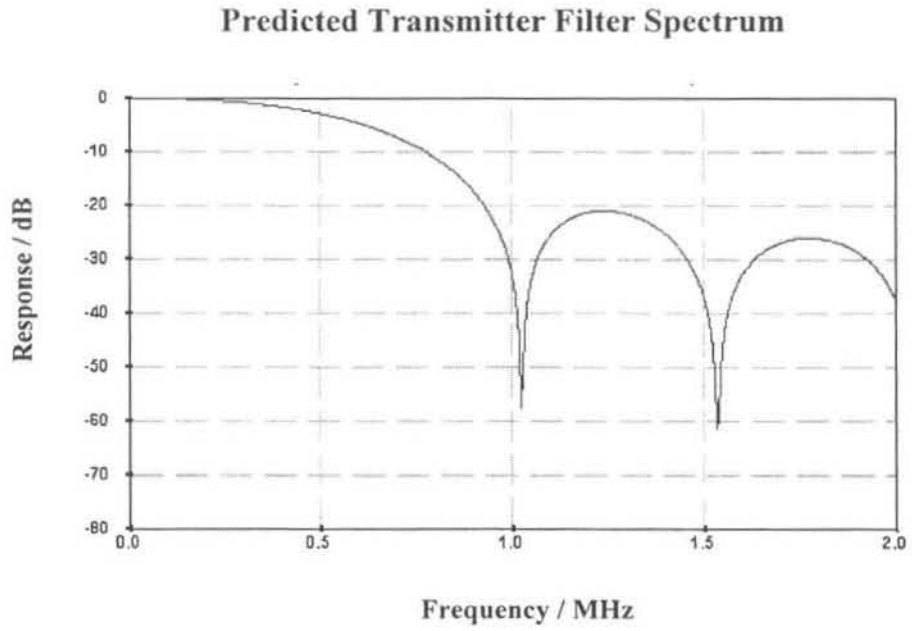

Figure 9.7 – Predicted Root-Raised Cosine Filter Response

Having established the basic communications parameters, the standard FIR calculation may be optimised.

9.6.2.1. Realisation Structure of the Transmitter Filter

There are many techniques available to the developer for implementing an FIR filter. These incorporate a plethora of structures relating each of the taps of the filter. Further, there are means of taking advantage of the signal nature that can further optimise the filter realisation. In the case of this project, what was initially a design flaw ultimately paved the way for further optimisation, since the nature of the input signal was more clearly understood. Initially, the transmitter filter was designed so that the input chip was present for each of the 8 samples. However, the following transmission/reception elements have to be considered:

$$S(f) \longrightarrow T(f) \longrightarrow C(f) \longrightarrow R(f)$$

The input to the detector is ideally raised cosine and is defined by the convolution of T(f) (the transmitter filter frequency response) and R(f) (the receiver filter frequency response). C(f), the channel response, distorts this, but S(f), the frequency characteristic of the data source, should not affect the overall system. If, however, S(f) is applied at the input to T(f) for each of the 8 samples, it causes the overall frequency response to broaden, occupying and wasting bandwidth. The solution is to allow S(f) to approach an impulse, which in itself allows for T(f) to be realised in an optimal fashion.

Using Interpolation For Optimisation Effectively, the transmitter filter increases the sample rate from 1.024MHz to 8.192MHz. This interpolating action may be translated to an optimised design. S(f), the source frequency response, approaches an impulse when only one out of the N transmitter input samples is set to the actual sample level, while the remaining N-1 samples are set to 0. Here, the larger N becomes, the more valid the approximation is. Since N-1 samples are effectively zero, the suggested realisation structure is as follows:

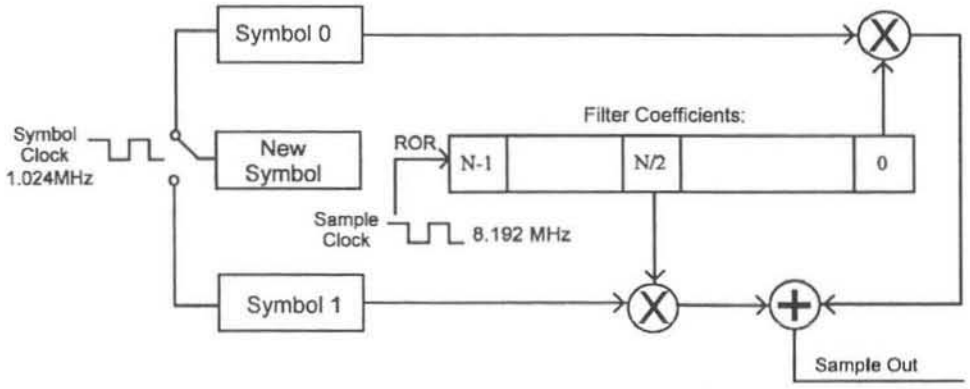

Figure 9.8 – Interpolation-Optimised Transmitter Filter Structure

From Figure 9.8, it can be seen that the N filter coefficients are rotated to the right at a rate of 8.192MHz. On a new symbol event (dictated by the 1.024MHz symbol clock), the appropriate symbol register is loaded with the new contents. Since there are 16 filter coefficients, and each symbol needs to be multiplied by each coefficient at least once, a symbol will remain loaded in a symbol register for 16 sample clock edges. However, a new symbol arrives every 8 sample clock edges. When this event occurs, the new symbol is loaded into the counterpart symbol register to preserve the current symbol register. In this manner, the numerous multiplications, where the multiplicand is actually 0, are avoided. Clearly, in terms of resources, there are only two multiplication entities and one addition entity associated with the transmitter filter. In addition, these entities are utilised only once on each sample clock edge, resulting in two multiplications and one addition per sample clock cycle.

The following table gives an example of a system where there are 4 filter coefficients

Table 9.1 – Example of Interpolation-Optimised Transmitter Maths

1.	0	$s_{-1}=0$	0	s_0
2.	h_3	h_2	h_1	h_0
3.	h_0	h_3	h_2	h_1
4.	0	s_1	0	s_0
5.	h_1	h_0	h_3	h_2
6.	h_2	h_1	h_0	h_3
7.	0	s_1	0	s_2
8.	h_3	h_2	h_1	h_0
9.	h_0	h_3	h_2	h_1

Table 9.1 gives a simple example to help explain the operation of the entity presented in Figure 9.8. In row 1 of the table, s_{-1} is set to zero, while a new symbol is loaded into s_0. In the line 2 of the table, it is seen that s_0 is multiplied by h_0. This occurs on the rising edge of the sample clock. On line 3, s_0 is multiplied by h_1. By line 4, there is a new symbol, but s_0 has only been multiplied with 2 of the coefficients, so it is preserved, while s_{-1} is replaced with s_1. On line 5, s_1 is multiplied with h_0, s_0 is multiplied with h_2 and the result accumulated to produce the output. On line 6, s_1 is multiplied with h_1, s_0 is multiplied with h_3 and the result accumulated to produce the output. By line 7, s_0 has obeyed the FIR filter requirements of being multiplied with each coefficient, and it can be dropped from the symbol register, to be replaced with s_2. This procedure will repeat indefinitely. This realisation structure has the results contained in Table 9.2.

Table 9.2 – Transmitter Filter Implementation Results

Summary of results for transmitter filter		
	Unsigned	**Signed**
Maximum sample clock rate	22MHz	63MHz
Maximum symbol clock rate	95MHz	95MHz
Logic cells	515	519

Table 9.2 shows a comparison between entities making use of signed and unsigned components. Section 5.4 introduced the idea of the "unsigned" bottleneck, and this table reflects those initial thoughts. While the signed entity is substantially quicker, it is also slightly larger. However, due to the minimal resource usage increase in the case of the signed entity, the trade-off between speed and size is not as critical as the issue of preserving types when interfacing the entity with other components. Due to the nature of the VHDL source defining this transmitter entity, the symbol clock is not actually used to clock a new symbol into the symbol register. Instead, on each sample clock, the state of the symbol clock is compared with its stored, previous state. If its previous state was 0, and its new state is 1, a new symbol has arrived, and it is loaded into the appropriate symbol register. So, while the clocks are actually derived from each other in terms of external test-bed conditions, the entity has no prior knowledge of this relationship. Therefore, the symbol clock has no fixed timing relationship relative to the sample clock within the entity.

9.6.2.2. Transmitter Filter Results

Frequency Domain

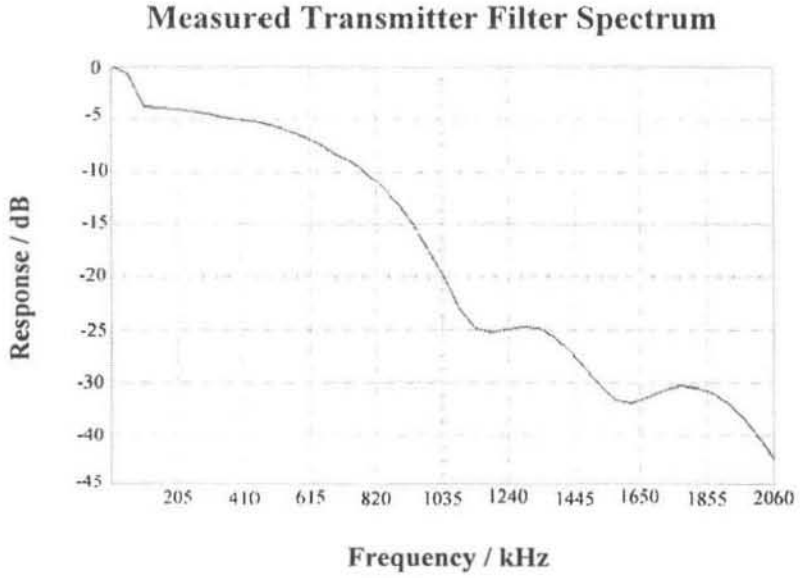

Figure 9.9 - – Measured Root-Raised Cosine Filter Response (Transmitter)

Figure 9.9 clearly shows the raised cosine response, as measured on the software radio DAC outputs. The predicted results are seen in Figure 9.7 and there may be some confusion over the apparent difference in performance in the stop-band when comparing the measured results with those predicted. The higher attenuation in this band is due to the post DAC filters built into the software radios. However, the cut-off frequency of approximately 1.024MHz is met.

9.6.3 Implementation of the Receiver Filter

In the case of the transmitter filter, the convolution occurred at 1/N times the sample rate. This is not true for the receiver filter that convolves coefficients and samples at the true sample rate. As a result, every input sample is involved in the generation of the output sample in a single cycle. This induces high resource usage, but by investigating two realisation structures, the system may be optimised.

9.6.3.1. Realisation Structures of the Receiver Filter

Transversal Filter

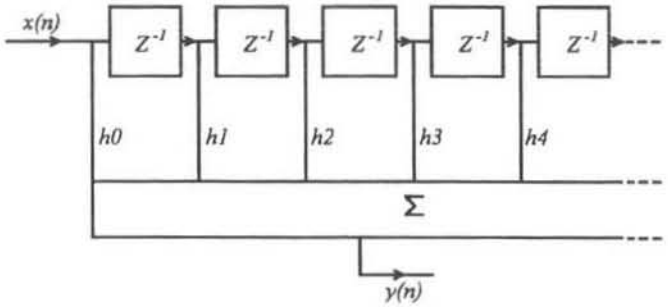

Figure 9.10 – Transversal Filter Implementation

In terms of implementing the direct FIR equation, this structure is particularly easy to realise. As can be seen in the figure, the samples are merely delayed in a lengthy shift register. These delayed samples are then multiplied by the coefficients. This implementation is particularly wasteful of resources and also incurs severe speed penalties. In the system under investigation, this structure was initially implemented in the following manner:

- The width of the input samples was a generic parameter of default value 12 (the bit-width of the ADC)
- The receiver coefficients were scaled to 10-bits
- Both the receiver coefficients and the input samples were stored in 160- and 192-bit shift registers respectively
- The registers were convolved in a loop of length 16, each being shifted the appropriate amount on each iteration

As was the case with the transmitter filter, two approaches were taken in implementation – those of using signed and unsigned maths. Table 9.3 reveals the comparative results.

Table 9.3 – Signed and Unsigned MAC Comparison

Performance Issue	Unsigned	Signed
Logic Cells	1352	1369
Maximum Clocking Frequency	9MHz	78MHz

The sample rate of the receiver filter is 8.192MHz. Table 9.3 shows that, while the unsigned receiver filter implementation is satisfactory, it is dangerously close to the critical system speed.

Transversal Receiver Filter Results

Frequency Domain Response

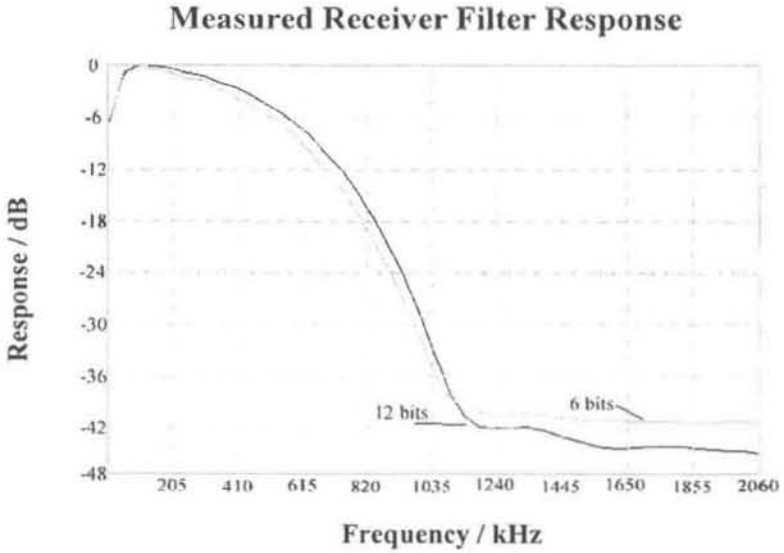

Figure 9.11 – Measured Root-Raised Cosine Filter Response (Receiver)

Time Domain Response With Noise

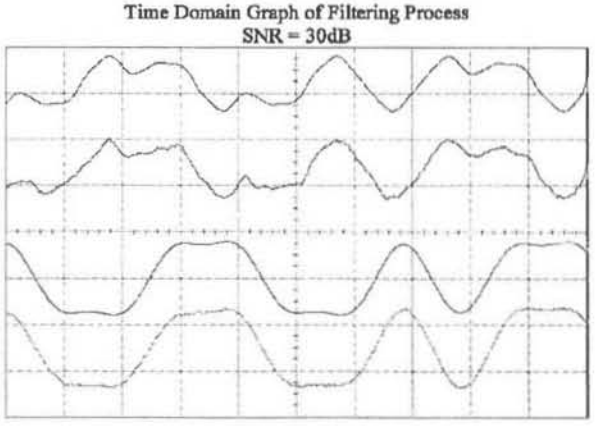

Figure 9.12 – Time Domain Response of Filter with SNR = 30dB

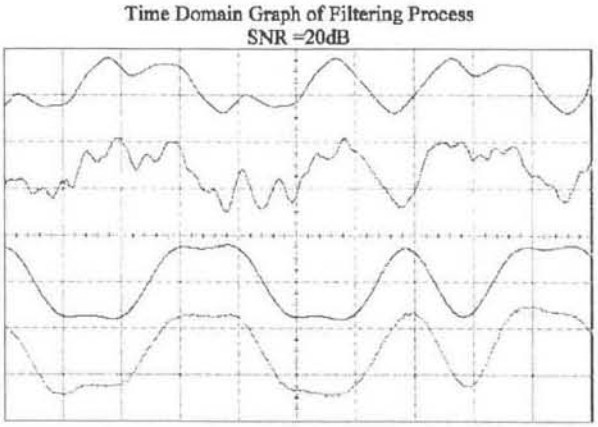

Figure 9.13 – Time Domain Response of Filter with SNR = 20dB

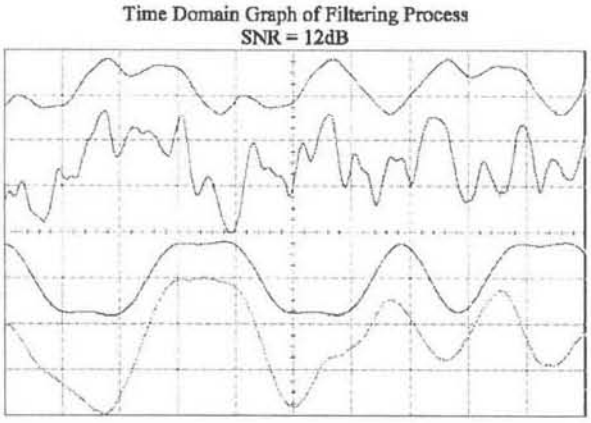

Figure 9.14 – Time Domain Response of Filter with SNR = 12dB

Figure 9.12, Figure 9.13 and Figure 9.14 show time domain graphs for varying Signal to Noise Ratios, showing how the receiver filter performs in the presence of AWGN. In each of these three graphs, the top waveform is the recorded transmitted signal. The second waveform is the transmitted signal with AWGN of a specific value (please see the chapter on simulating channel distortions in the FPGA - Chapter 11 - to see how this was achieved). The third waveform is the ideal (no noise) received waveform, while the bottom waveform is the input to the detector after receiver filtering.

Linear Phase Structure

The transversal filter is wasteful because it does not take advantage of the structure of the FIR coefficients. For the current system, Table 9.4 shows the multiply and accumulate relationships that exist between the coefficients.

Table 9.4 – FIR Linear Phase Structure Coefficient Comparision

Coefficient	Sample 1	Sample 2
[h0]	$[x_n]$	
[h1]	$[x_{n-1}]$	$[x_{n-15}]$
[h2]	$[x_{n-2}]$	$[x_{n-14}]$
[h3]	$[x_{n-3}]$	$[x_{n-13}]$
[h4]	$[x_{n-4}]$	$[x_{n-12}]$
[h5]	$[x_{n-5}]$	$[x_{n-11}]$
[h6]	$[x_{n-6}]$	$[x_{n-10}]$
[h7]	$[x_{n-7}]$	$[x_{n-9}]$
[h8]	$[x_{n-8}]$	

As can be seen in Table 9.4, there is a high degree of symmetry in the coefficient vector structure. The realisation seen in Figure 9.15 takes advantage of this fact:

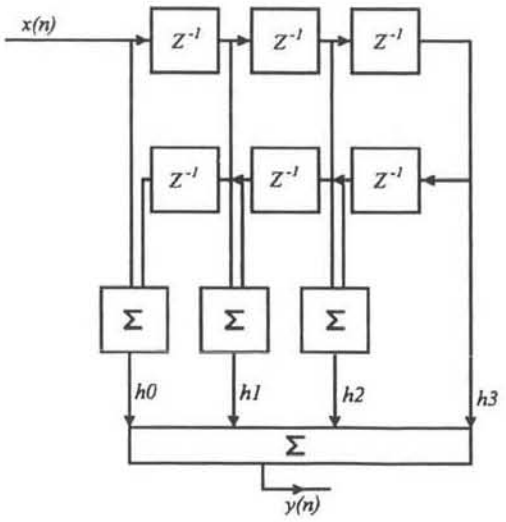

Figure 9.15 – Linear Phase Structure for Filter

As opposed to the transversal method that has N multiplications and N-1 additions, this method has only N/2 multiplications and N additions. This produces considerable savings in both speed and resource usage, while preserving the spectral efficiency of the receiver filter.

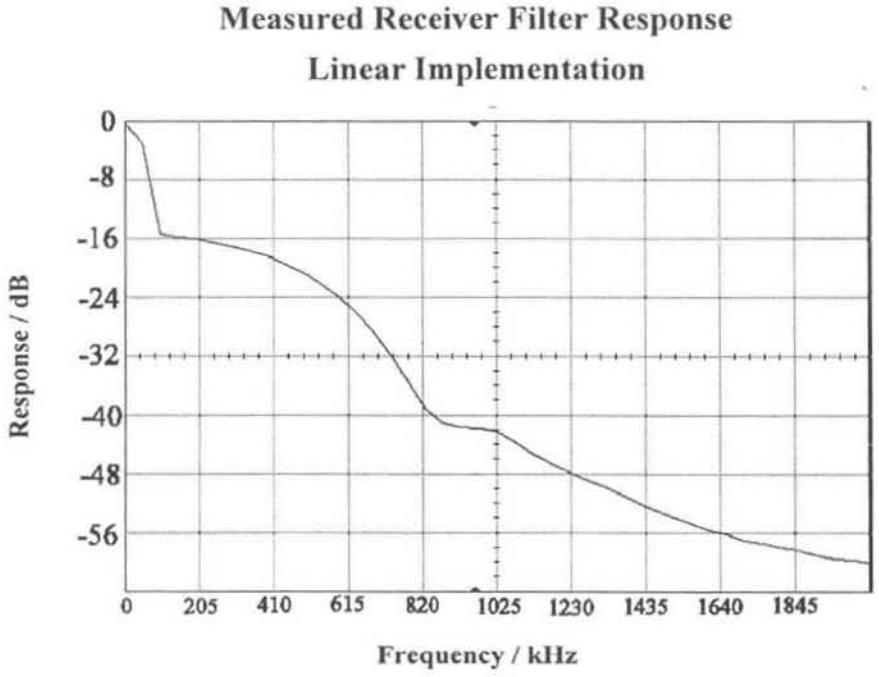

Figure 9.16 – Measured Filter Response for Linear Phase Structure (Receiver)

Summary of Resource Usage and Speed

Table 9.5 – Comparison of Receiver Filter Realisation Resource Usage and Speed

Realisation	Logic Cells	Gain (%)	Speed (MHz)	Gain (%)
Unsigned Transversal	1352	–	9	–
Unsigned Linear	1012	25	12	33
Signed Transversal	1369	–	78	–
Signed Linear	1025	25	78	0

Partial Summation Parallel Computation Structure

Distributed, or parallel, arithmetic is a further means of optimising the memory usage and allowing the system to run at a higher speed. While this entity has been implemented to allow for measured results, it has not been included in the final system. It does, however, allow for an insight into further optimising the system. There are many sources of information regarding this new area of implementation research, most notably the manufacturers of FPGA devices themselves. It therefore comes as no surprise that Altera, the manufacturers of the FPGA devices used in this project's software radio, have published an article on the theory of operation. This may be found in their document entitled "Implementing FIR Filters in FLEX Devices", [19].

Theory

Vector-based multiply and accumulate operations have their basis in fundamental logic multiplication and can be embodied in a look up table. The process of splitting up the multipliers and generating the address into the look up table is best demonstrated with an example.

Consider a situation in which 4 binary-represented numbers will be multiplied and accumulated. Each number is a 2-bit number. Table 9.6 records these numbers.

Table 9.6 – Distributed Arithmetic Multiplier and Multiplicand Breakdown

Multiplicand	Multiplier
01	10
11	11
10	01
01	11

This has the equivalent multiplication of:

Multiplicand		01		11		10		01	
Multiplier	x	1**0**		1**1**		0**1**		1**1**	
LSB result		00	+	11	+	10	+	01	= 110
MSB result		01	+	11	+	00	+	01	= 101
									10000

If the MSB result is shifted once to the left and added to the LSB result, in the horizontal direction, and then the result from each of these partial products is then added, the result (10000) is identical to the actual MAC performed between all 4 sets of numbers. At this point, the concept of corresponding bits needs to be expounded. Simply, the corresponding bits in two or more binary represented numbers hold the same significance. For example, in the example given, the corresponding bits for bit 0 of each of the multipliers form the number (0111) when concatenated (the enlarged digits in the multiplier row form this number). Notice how, if the corresponding placeholder has a binary value of 1, then the corresponding multiplicand is included in the summation, otherwise it is rejected. In this manner, if we take corresponding bits in successive binary numbers, we can form an index into a lookup table. The above information in the lookup table may be generated as seen in Table 9.7.

Table 9.7 – Partial Summation According to Corresponding Bits

Corresponding Multiplier Bits	Result
0000	00 + 00 + 00 + 00 = 000
0001	00 + 00 + 00 + 01 = 001
0010	00 + 00 + 11 + 00 = 010
0011	00 + 00 + 11 + 01 = 011
0100	00 + 10 + 00 + 00 = 011
0101	00 + 10 + 00 + 01 = 100
0110	00 + 10 + 11 + 00 = 101
0111	00 + 10 + 11 + 01 = 110
1000	01 + 00 + 00 + 00 = 001
1001	01 + 00 + 00 + 01 = 010
1010	01 + 00 + 11 + 00 = 011
1011	01 + 00 + 11 + 01 = 100
1100	01 + 10 + 00 + 00 = 100
1101	01 + 10 + 00 + 01 = 101
1110	01 + 10 + 11 + 00 = 110
1111	01 + 10 + 11 + 01 = 111

The exact same look up table may be used for every corresponding bit in the multipliers. However, the significance of the bit determines the look up output. In the example given, for the corresponding MSB's of the multipliers, the index is (1101) and the partial result is 101. However, this result needs to be shifted once to the left before being added to the information stored at address (0111), which is (110). Therefore the bit position determines the shift extent before addition. The problem with such a system is that the size of the LUT is an exponential function of the number of multipliers. ($A = 2^n$ where A is the number of address spaces and n is the number of multipliers). There is a simple solution – to move some of the pre-calculated additions to outside the LUT and split the LUT into two entities, as seen in Table 9.8.

Table 9.8 – Splitting the Look-Up Table for Resource Saving

LUT1 **LUT2**

Corresponding Multiplier Bits	Result	Corresponding Multiplier Bits	Result
00	00 + 00 = 000	00	00 + 00 = 00
01	00 + 11 = 011	01	00 + 10 = 10
10	01 + 00 = 001	10	01 + 00 = 01
11	01 + 11 = 100	11	01 + 10 = 11

With the scheme recorded in Table 9.8, the required number of address spaces is halved, while the size of the storage element in the second LUT is diminished by one bit. However, once each of the look up tables returns its stored data, these partial summations need to be added together, incurring additional (and far slower) logic. Once again, the trade off between speed and resource usage is prominent.

Implementation of the Partial Summation Parallel Computation Structure

The theory of vector based MAC's incorporating LUT's can be applied to FIR filter design. Here, the multiplicand is an element in a set of static coefficients – h(n), while the multipliers dynamically generate the index into the LUT – x(n). For the system under consideration, the multipliers are 12-bit unsigned numbers while the LUTs contain the partial of the correctly scaled coefficient summations. A MATLAB function was developed to generate the look up table data. In developing such a function, the following parameters are important:

- The number of coefficients
- The number of samples per symbol
- The number of look up tables to generate

The proposed system uses 16 coefficients, 8 samples per symbol. At this point, a conceptually difficult process needs to be developed, and it actually represents the key to optimising the system – the addressing scheme.

The Addressing Scheme

Essentially, the addressing scheme is a dimension transpose performed on the input data. Consider the system where the input sample is B bits wide and there are N samples. The system would then actually consist of B look up tables (requiring the construction of B addresses - each N bits wide). This dimension transpose embodies the speed and resource bottleneck for the system. The first option is to create a two-dimensional array, whereby the current sample is broken into its B constituent bits and then stored in B shift registers. These registers can then act as the addresses. However, describing such a system in VHDL produces poor results in terms of entity size. A second option would be to store the incoming samples in a long shift register of size (B x N) bits. Then, address 0 would be constructed using every B^{th} bit, starting at bit 0, and generally, address i will be constructed using every B^{th} bit, with the offset from the start of the register being i bits. Such a system remains the work of a further investigation. The system would be constructed as follows:

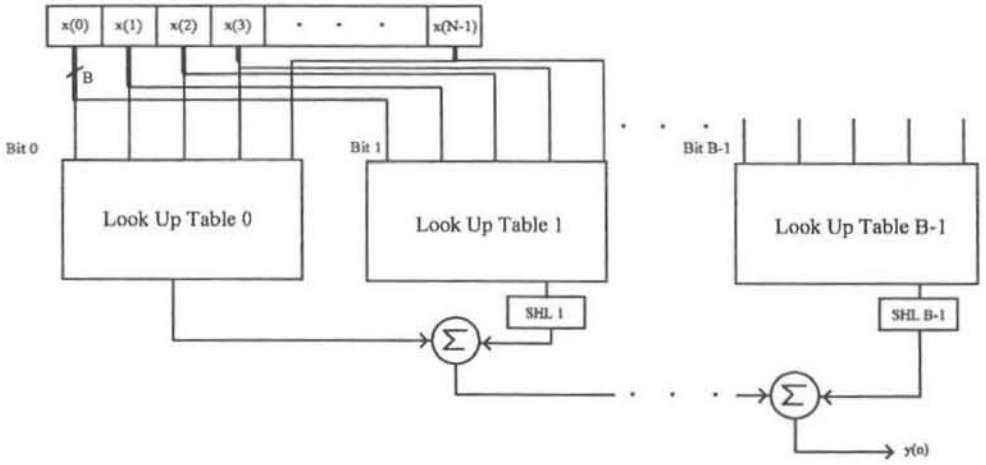

Figure 9.17 – Parallel Arithmetic Look-Up Table Integration

In the diagram, SHL is a logical left shift.

VHDL Considerations

Important in this design is the description of the processes in VHDL. It must be noted that VHDL places some fairly strong limitations on the implementation. VHDL is a strongly typed language. Therefore, in Figure 9.17, the inputs to each of the adders must not only be of the same type, but they must also have identical bit-widths. So, for a system consisting of N samples, each made up of B bits, the final LUT result will have to be shifted by B-1 bits. To add this successfully to the other LUT results, all of the LUT's should have the same output width. Thus, the shift and the width preparation are performed inside each LUT. This increases the resource usage.

Results - Resource Usage and Speed Performance

A fairly primitive version of the parallel look up table scheme has been implemented to test its effectiveness. The LUT's were optimised as combinational logic entities, making use of the select operation defined in VHDL as follows:

```
with choice_expression select
   target <= ( expression when choices, )
             expression when choices;
```

Here, `choice_expression` is the address that will be used to return the partial summation (target). The returned data (expression) is dependant on the input address (choices). Essentially, this embodies a large truth table that can be optimised as a combinational logic entity. However, using this scheme, one complete look up table utilises 105 logic cells (2.1% of the FPGA – including shifting and addition operations).

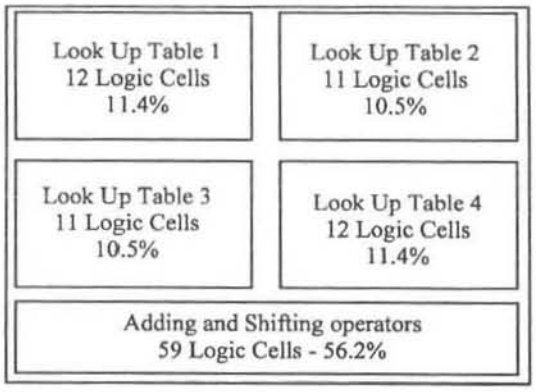

Figure 9.18 – Resource Usage for Look-Up Table with Four Elements

To optimise the resource usage is fairly complex. First, the trade-off needs to be fully understood. Each constituent look up table (there are 4 in the above example) incorporates partial summations. These results then need to be summated – in the above example, more than half of the logic resides in this operation alone – to produce the final data. The trade-off comes in knowing how to balance the logic use between the constituent look up tables and the addition logic. If the number of look up tables is decreased, their size increases exponentially as a power of 2, while the addition logic decreases linearly. Knowing the optimal trade-off point is probably best achieved using trial and error, although a more complete analysis of the gate usage may be tackled. For example, halving the number of constituent look up tables in the above example produces the following entity:

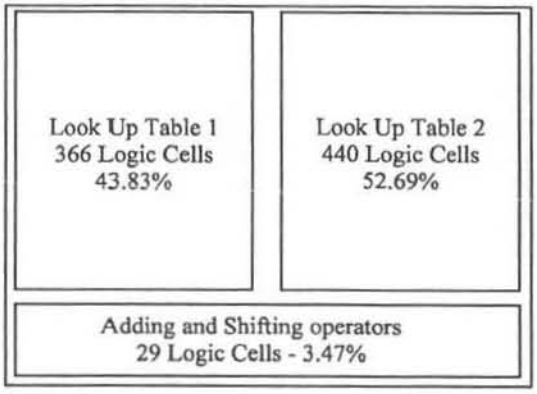

Figure 9.19 – Resource Usage for Look-Up Table with Two Elements

The comparative results for the above look up table are less than desirable, as seen in Table 9.9.

Table 9.9 – Comparison of Look-Up Table Size and Resource Usage/Speed Analysis

Look up Table Type	Logic Cells		Speed	
4 Constituent Tables	105		28MHz	
2 Constituent Tables	835	+695%	19MHz	-32%

Clearly, there is an optimal point beyond which the LUT will not decrease in size. A general rule to follow, which can be seen in action in the above example, is to attempt to create a LUT where the addition logic and the collective truth table logic have near-identical sizes. Suppose that such a state can be reached. It can be seen that this must be near optimal because the following two steps will only serve to worsen the situation:

- Doubling the number of truth tables will decrease their constituent size. However, according to the linear addition rule, the logic required to implement the addition will also double. Since the addition logic is already 50% of the LUT, and will now be doubled, this must have a negative effect on the resource usage.
- Halving the number of truth tables will serve to exponentially increase the contribution to the system size afforded by the truth tables. Thus, while the truth table logic is originally 50%, this will be multiplied many times over, having a negative effect on the resource usage.

In the example given, the LUT with 4 constituent LUTs has 56.2% of the logic use in the addition, and 43.8% of the logic use in the truth tables. This obeys the 50% rule well, and is probably near optimal.

Resource Enhancements – Partial Serial Computation

One particular means of optimising the system lies in reduced parallelism or resource re-usage. It may be noticed that each LUT contains exactly the same information, the only difference appearing in the shifting operation. One solution to this apparent redundancy is to reuse the look up tables in a serial fashion. True serial computation would take the following form:

There are N samples, each having B bits. In a purely serial implementation, the resulting B computations would all take place using a single look up table. However, instead of using a single clock edge, the system would use B clock edges. On the first clock edge, the LSB corresponding bits would form the input address, and the result would be stored. Thereafter, on each successive clock edge i, the result will be shifted by i bits to the left before being accumulated in the storage register. After B clock edges, the storage register can be latched to the interface register and cleared for the next clock edge. This system has the advantage of reducing the resource usage, but it will severely affect the speed. The solution is to use a hybrid

scheme wherein there is a degree of parallelism, while making use of the resource saving offered by the serial reduction.

9.7 Conclusion

Base-band pulse shaping presents a challenge for the FPGA designer. This chapter has explored several options for implementing Finite Impulse Response filters to reduce bandwidth while maximising signal to noise ratios at the input to detectors. Ultimately, the speed enhancements offered by the FPGA make it a very attractive alternative to the traditional DSP-based filtering systems. However, the emphasis is really on the designer, where the task of implementation optimisation may prove quite daunting.

Chapter 10 MMSE Detection

The implementation of a Minimum Mean Square Error (MMSE) multi-user detector on an FPGA raises further difficulties. Chapter 4 explored the theory of resource partitioning in a software radio context. However, since the flexible radio platform used for implementation purposes in this project has only an FPGA base, resource partitioning is not a viable option for the developer. This chapter will examine the very specific case of the implementation of an MMSE detector for a downlink. Making use of Walsh codes and their desirable cross-correlation properties, the chapter will examine techniques for pilot channel symbol pattern selection and subsequent synchronisation. Further, this chapter will present an area where future work may be carried out to improve the system functionality. This will involve the partitioning of resources across the RS-485 link to the DSP subsystem. This link was never designed to carry off-line processing information, but it does offer the developer one means of performing the mathematically intensive MMSE operations on a DSP.

10.1 The MMSE System Model

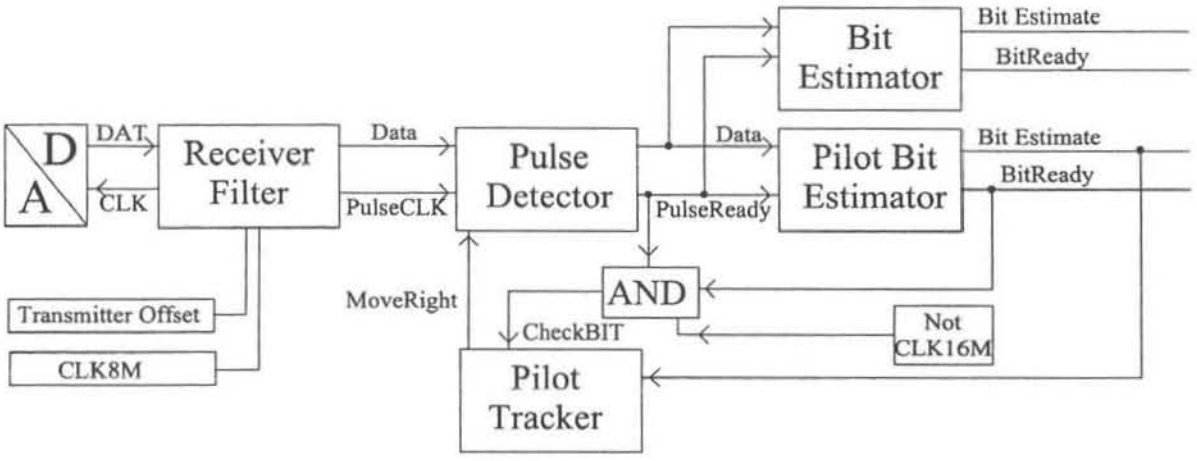

Figure 10.1 – The MMSE Pilot Synchronisation Diagram

Figure 10.1 shows the essential elements of the system receiver. The A/D is clocked at 8.196 mega samples per second. This corresponds with the receiver filter characteristic of 8 samples per symbol, the base symbol rate being 1.024 mega symbols per second. Chapter 9 deals with the characteristics of the receiver filter, and how the sample rate is related to the symbol detection. Apart from the 8.192 MHz clock, the Receiver filter is also fed the transmitter offset, since the A/D supplies it with a 12-bit unsigned number. Thus, the receiver will convert the sample to a 12-bit signed number (by subtracting the transmitter offset) before processing. On completion of each convolution of the sample with the filter impulse, the receiver filter will generate a rising clock edge indicating the presence of a new pulse. This data is fed to the pulse

detector. In this fairly simple detector, a counter will keep track of the number of pulses that have been generated by the filter. On start-up, the pulse detector assumes that the data from the receiver filter represents a sample in the middle of a chip (symbol). It will send this "pulse estimate" to the MMSE structure. The procedure thereafter follows a repetitive pattern. The pulse detector keeps track of the number of samples clocked in from the receiver. Once the counter reaches a value of 8, it will send the next filter sample through to the MMSE structure. Thus, every 8^{th} sample from the filter is shipped on to the Bit Estimators in the MMSE structure. Initially, it is highly unlikely that the receiver will be in chip synchronism. As a result, the MMSE structure will collect 16 chips (representing a single bit) and attempt to produce the corresponding bit estimate, accompanied by a 64kHz clock (BitReady). To aid in synchronisation, a specific bit sequence, with a specific code, is sent as a pilot signal. The pilot tracker has the task of monitoring the bit sequence state, and if a bit estimate does not match the required next bit state, it takes corrective action. It achieves this by sending a "move right" pulse to the pulse detector. This pulse is effectively a single chip delay pulse, and it implements a sample "scanning" action. Each time the signal is pulsed, the pulse detector will delay the increment of its pulse counter. As a result, the detector will scan from left to right until it synchronises with the peaks of the chip pulses.

10.2 Timing Analysis

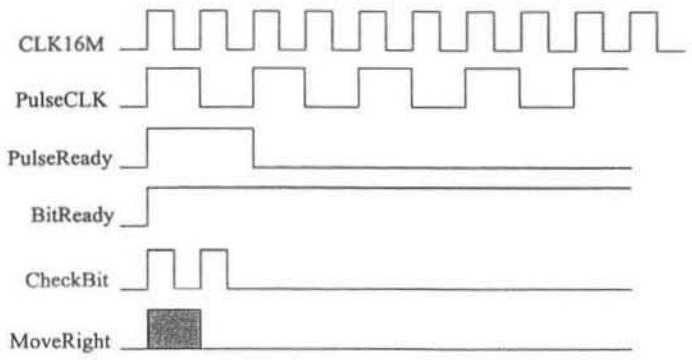

Figure 10.2 – Timing Diagram for Pilot Synchronisation

Figure 10.2 shows the timing diagram for a synchronisation period. It shows the point where the bit estimate is ready. The CheckBit pulse has two purposes. First, it will trigger the comparison of the received bit with the expected bit. Based of the outcome of this comparison, the scanning signal MoveRight will either be set (if there is an error) or it will remain clear. This all happens on the first rising edge of CheckBit. Second, it will clear MoveRight once the system is waiting for the next bit, to prevent a continual scanning. This process will synchronise under all phase differences between the transmitter and the receiver.

10.3 Performance Analysis

Analysing the system reveals the following synchronisation delays:

- The Pilot bit sequence consists of 8 bits
- Each bit consists of 16 chips
- Each chips consists of 8 samples

Using an exhaustive synchronisation procedure in MATLAB, which cycles through all of the possible phase differences (in chips) between the receiver and transmitter, produces the results seen in Figure 10.3.

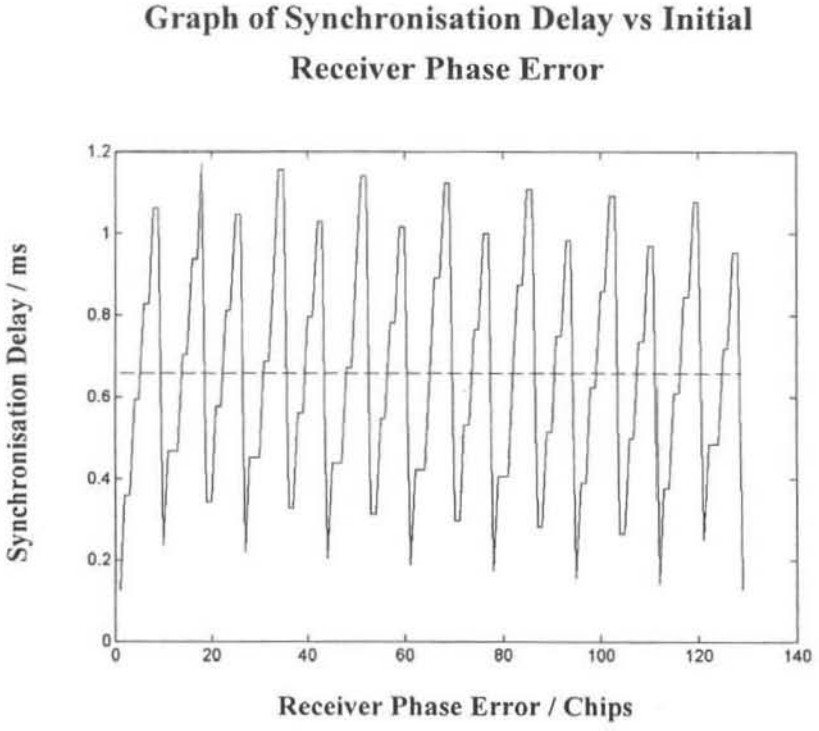

Figure 10.3 – Synchronisation Delay for Specific Initial Phase Error

Figure 10.3 shows the performance of the synchronisation process. As the receiver phase error varies from 0 chips through the full 128 chips, so the system is subject to a fairly unpredictable synchronisation delay. The mean, however, is 0.6589ms, corresponding to the loss of 42 bits during synchronisation.

10.4 Choice of Pilot Signal Code and Bit Sequence

The procedure described so far operates successfully only if two imperative parameters are carefully designed. These are:

- The Pilot signal code
- The Pilot sequence

If the pilot signal is to be successfully de-spread at the receiver, the pilot Walsh code has to be carefully chosen. First, it must be of maximal length. If it is not, and it has a sub-pattern that repeats within its length, then the bit pattern may be successfully retrieved, while the tracker is actually out of phase by an integral number of chips. This obviously distorts the information decoded on other channels. A good example can be seen when using Walsh code 0 to encode the pilot channel. Analysing the data on code 1 will reveal that 50% of the time, the inverse of the expected data is received.

Second, the pilot bit sequence needs to be well designed. Any sequence where a sub-pattern repeats itself has a non-zero probability of failing, since the tracker may latch to a pattern offset from the start of the bit sequence. A guideline for designing this bit sequence is as follows:

- Choose a non-repeating bit sequence half the length of the total bit sequence
- Reverse the order of the bits
- Invert the reversed bits
- Append the new sequence to the end of the old sequence to make up the complete sequence

This technique avoids the pitfalls of sequence patterns and allows the MMSE detector to operate correctly.

10.5 Implementation Architecture

This particular implementation takes advantage of the fact that the downlink can use Walsh codes. This is because the base station can ensure perfect synchronisation between users. As a result, a fairly simple architecture may be used to detect the user channels. The mathematical treatment of this technique may be found in section 3.4, where the Walsh code case reduces to a fixed-coefficient implementation. Here, the accumulated chips are multiplied by a constant where the sign of the constant is derived from the original chipping matrix (see Equation 3.9). This allows for accurate performance (see section 11.6 for results) in spite of not incorporating noise as a factor in the matrix inversion. This success is due primarily to the matched receiver filter that maximises the SNR at the input to the detector. Many of the other issues broached in section 3.4 can also be incorporated in the MMSE implementation architecture. The implementation issues surrounding some of the entities outlined in Figure 10.1 will be expanded on in the following sections.

10.5.1 The Pulse Detector Entity

The following code fragment helps to explain how the system is able to scan either to the left or the right to aid pilot synchronisation:

```
12.     UpdateCounter : process(PulseCLK,Synchronised)
13.     begin
14.        if (Synchronised = '0') then
15.           Counter    <= "000";
16.        elsif (PulseCLK'event and PulseCLK = '0') then
17.           if (MoveLeft = '1') then
18.              if (Counter < "110") then
19.                 Counter <= Counter + "010";
20.              else
21.                 Counter <= Counter - "110";
22.              end if;
23.           elsif (MoveRight = '0') then
24.              if (Counter = "111") then
25.                 Counter <= "000";
26.              else
27.                 Counter <= Counter + "001";
28.              end if;
29.           end if;
30.        end if;
31.     end process;
```

The pulse counter is updated by an 8 MHz clock signal. As already stated, only every 8^{th} sample is sent to the bit estimator. Consider the case where the symbol peak is actually one sample to the right of the current sample:

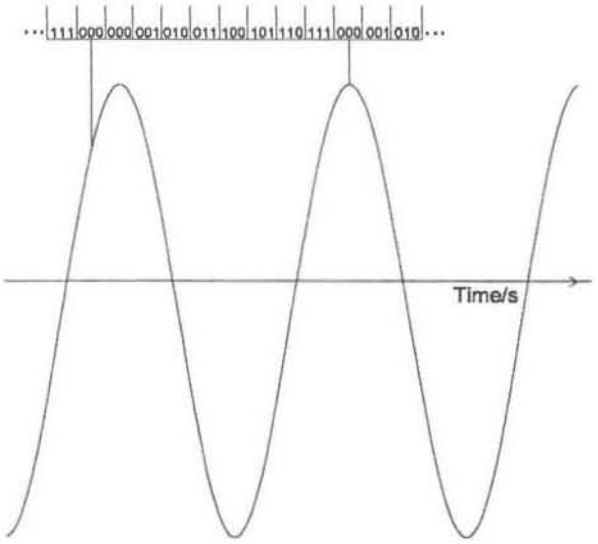

Figure 10.4 – Sample Offset to the Left

Figure 10.4 shows a simplified graphical representation of successive symbols. The 3-bit number on the figure encodes the symbol counter. When the symbol counter attains a value of zero, the current sample value is sent to the bit-estimator as a symbol peak. In Figure 10.4, the first peak estimate is displaced one sample to the left of its correct position. As a result, the system will attempt to scan to the right so that the zero count will occur at the true position of the next peak. Lines 23-29 of the previous code fragment achieve this by simply not updating the symbol counter on a "MoveRight" condition. As a result, the symbol counter will be zero for two samples in succession, allowing the counter to return to zero at the next true peak, shown in Figure 10.4. A similar situation occurs when the true symbol peak is one sample to the left of the estimated point:

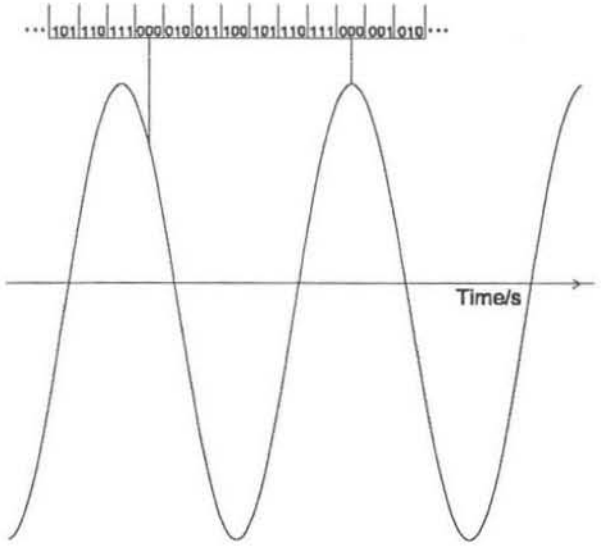

Figure 10.5 – Sample Offset to the Right

Figure 10.5 shows the case when the pulse detector sends a sample to the bit estimator that is actually to the right of the true symbol peak. As seen in the previous code fragment, lines 17-22, if the "MoveLeft" condition is set, then a value of 2 is added to the current symbol counter. As a result, on the next clock pulse, the symbol counter will effectively have skipped over a sample so that, when it returns to zero, the sample it reports will be the true symbol peak, as seen in Figure 10.5. The pulse detector then passes the sample on to the bit estimator, which is described next.

10.5.2 The Bit Estimator Entity

The bit estimator entity is responsible for convolving the pulses from the pulse detector with the rows of the fixed MMSE filter matrix. The following code fragment shows the decision point of the estimator:

```
1.          if (SymbolCount = "1111") then
2.             if (tmpAccumulate <= "010000111110110000000000") then
3.                BitEstimate <= '0';
4.             else
5.                BitEstimate <= '1';
6.             end if;
7.             BitReady       <= '1';
8.             tmpAccumulate  := "010000111110110000000000";
9.             SymbolCount    <= "0000";
10.           else
11.              SymbolCount    <= SymbolCount + 1;
12.              BitReady       <= '0';
13.           end if;
```

In line 8 of the previous code fragment, the accumulator is loaded with the necessary offset. This value corresponds to the maximum attainable value for the correlation of the expected samples with the fixed MMSE filter values. Line 1 shows that, after 16 symbols have been convolved and accumulated, the accumulator is once again checked against the initial offset. If it is less than or equal to this value, then the bit is estimated to be zero, otherwise it is reported as being a one. Once the bit is ready, the "BitReady" signal is pulsed high.

10.5.3 The Pilot Tracker

As it stands, the implementation of the pilot tracker is very simple. The values for the pilot symbol sequence as seen in 10.4 are expected at the receiver on Walsh code 16. If the successive pilot symbols are not correctly detected, then the tracker will force a move to the right for the pulse detector. This will continue until the pilot sequence is correctly extracted.

10.5.4 Future Work

It is desirable that a true MMSE filter be implemented, one capable of detecting the uplink information in addition to the downlink. This requires additional resource usage that the FPGA cannot provide. However, future work may investigate the use of the PC-based DSP card for performing off-line calculation of channel parameters. Thus, matrix inversion may be performed in the DSP and not within the limited FPGA sub-system. However, the 2 mega bit/second link to the DSP does not provide sufficient bandwidth for high-speed calculation in the event of rapid mobility or quickly changing channel parameters. Instead, this solution would be adequate for low-speed movement but would provide a more robust and generalised means of implementing the MMSE detector.

10.6 Conclusion

Multi-user detectors have many implementation difficulties, primarily due to their complex mathematical nature. This chapter presented a simple means of using a fixed-coefficient detector in conjunction with Walsh codes and a pilot synchronisation channel. The performance of such a detector, especially in the light that it detects all the users in a channel simultaneously, warrants further investigation into more advanced implementation techniques. However, this project has shown that the FPGA system is capable of implementing a low BER Multi-User Detector, and has many promising applications.

Chapter 11 Simulating Channel Distortion in the Software Radio

The testing of a communications system involves subjecting the channel to typical operating conditions. In the case of a test bed, it may not be easy to create the correct environment to carry out a complete performance analysis. The flexibility of the software radio allows for the simulation of many distortion phenomena, and, using software, these may be dynamically specified at the time of testing. This chapter will examine the reproduction of two common types of wireless channel distortions – uncorrelated Rayleigh fading and Additive White Gaussian Noise. The introduction of these two factors in the software radios allow the developer to distort the relevant signals and provide an estimate of the Bit Error Rate of the system. This chapter will also examine the building blocks used to generate a high-speed simulated environment incorporating Rayleigh fading and AWGN.

11.1 System Overview

Before the derivation and implementation issues are examined, it is important to establish the context within which the simulation will occur. This includes creating an understanding of the resource sharing techniques employed to optimally determine where the distortion types are generated. Figure 11.1 aids the explanation:

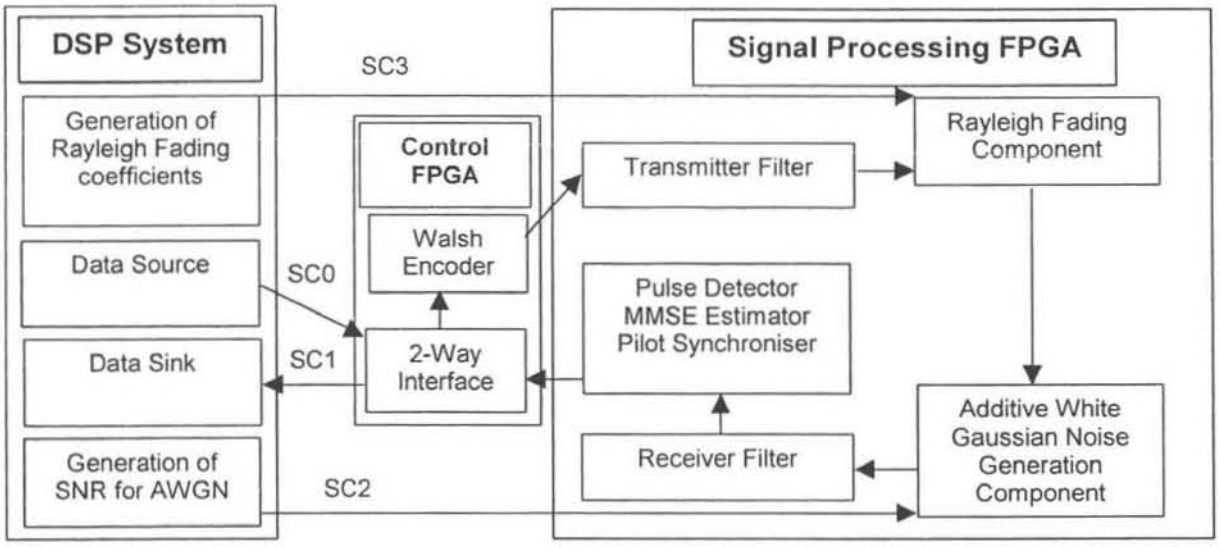

Figure 11.1 – Simulation System Overview

Figure 11.1 gives a new perspective on the system overview, modifying that seen in Figure 6.2 to include 4 high-speed serial streams for the purpose of simulation. SC0 is one 2 Mbps stream,

acting as a link to a data source, while SC1 receives information from the software radio and sends it to the data sink. SC2 and SC3 are also 2Mbps serial streams porting AWGN and Rayleigh fading information respectively. Further, as seen in Figure 11.1, both the DSP system and the FPGA are utilised in the distortion generation procedure. However, due to the differing complexity of the types of distortion involved, and the limited FPGA resources, a careful consideration of the resource sharing is needed.

Rayleigh Fading Source

The generation of the Rayleigh fading coefficients involves several computationally expensive manipulations. Primarily, these are used to transform a sample space from one statistical distribution to another. As the theory is developed, so it will become apparent that the FPGA is not the ideal resource to use for the purpose of generating the Rayleigh Fading coefficients. Further, utilising the DSP system allows for improved accuracy in the final coefficients. However, the DSP is not quick enough to produce a new coefficient within a single transmit interrupt. As a result, the coefficients are pre-calculated and stored in memory, being sent to the software radio on each transmit interrupt. These are then multiplied with the relevant signal within the FPGA to produce a Rayleigh-faded signal.

AWGN Source

The generation of the Additive White Gaussian Noise samples is computationally less burdensome than the Rayleigh fading. As a result, the FPGA is suited to high-speed generation of these samples. However, the calculation for converting the desired Signal to Noise Ratio to a coefficient that the FPGA can make use of is still performed in the DSP due to the complexity of its nature. As is the case with the Rayleigh Fading coefficients, this parameter is sent to the software radio over the SC-BUS. The generated samples are added to the relevant signal to produce a noise-degraded output.

11.2 Specifying Uncorrelated Rayleigh Fading

A communications channel subject to the effects of Rayleigh fading will experience distortion in the amplitude and phase of a transmitted signal. This is due to multi-path effects. However, to simplify the simulation of such a distortion, only the amplitude response will be considered. The following equation describes the actual Rayleigh distribution:

$$f(r) = \begin{cases} \frac{r}{\alpha^2} e^{-r^2/2\alpha^2} & 0 \le r \le \infty \\ 0 & r < 0 \end{cases}$$

.....Equation 11.1

Equation 11.1 generates the graphical distribution seen in Figure 11.2

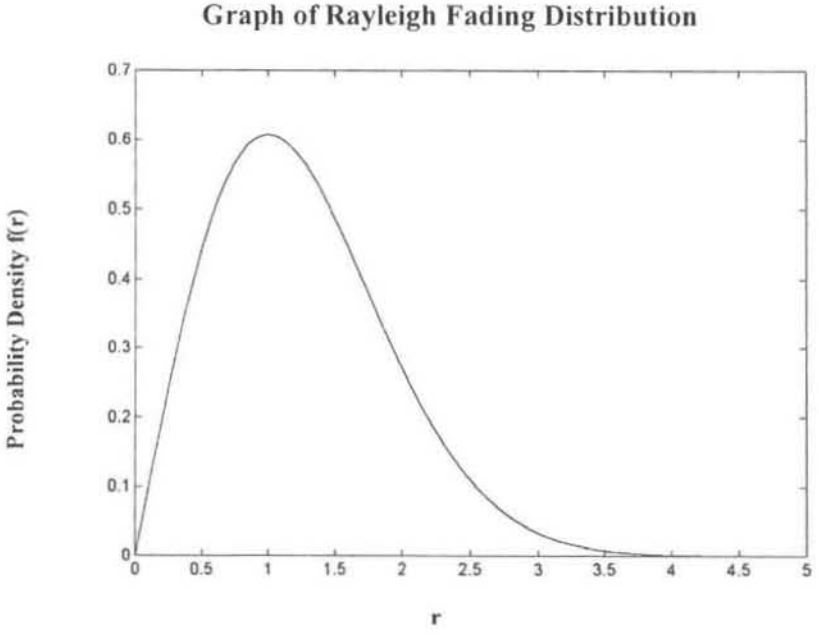

Figure 11.2 – Rayleigh Fading Distribution

Two common statistical distributions will be used to generate the Rayleigh fading coefficients – the uniform distribution and the normal (Gaussian) distribution. Before an examination of the generation of Rayleigh random variables can be undertaken, it is necessary to be able to generate Gaussian random variables from a uniform distribution. As seen in the system overview, the DSP system is used to generate the Rayleigh fading coefficients. It is therefore necessary to generate the Gaussian distributed samples on the DSP card, which is programmed in ANSI C. Unfortunately, ANSI C does not define a standard function for returning samples from a normal distribution. It does, however, have a standard uniform number generator that is can be used to create the Gaussian random deviate.

11.2.1 Generating Gaussian Random Samples

There are many techniques available for transforming a uniform random sample space to a normal random sample space. However, the developer has to be constantly aware of the target implementation platform. This is because the choice of generation technique usually involves a trade-off between the speed of generation and the accuracy of the generated samples. For this

first generation technique, speed is not considered, while a premium is placed on the final accuracy. This is because the samples are being generated in the DSP, where a plethora of standard ANSI C maths functions allow for accurate mathematical manipulation of variables. However, the generation of a single sample within the time frame of a transmit interrupt on the SC-BUS is not possible, since the DSP is not quick enough. Instead of this approach, a sufficiently large number of samples from the Rayleigh random space are stored during an initialisation period. These are then retrieved sequentially during processing, and sent to the software radio.

The Gaussian Distribution

[A. Ralston, 1967], in his work on mathematical methods as implemented on digital computers, [20], states that the unit normal (Gaussian) probability distribution is defined as:

$$R_U = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{R_n} e^{-x^2/2} dx$$ **.....Equation 11.2**

- R_U is a normalised uniform random variable
- R_N is a normalised normal random variable

In this case, there is no close-form solution for R_N and any technique must rely on an approximate generating equation. To this end, and with accuracy being foremost, there is a means of generating a Gaussian random variable given two uniformly distributed random variables:

$$R_N = \sqrt{-2\ln R_U}\cos(2\pi \cdot R_{U+1})$$ **.....Equation 11.3**

- R_U and R_{U+1} are the two uniformly distributed random variables
- R_N is the normal random deviate

To test the success of this approach, MATLAB was used to generate a sample space of 20,000 uniformly distributed random variable samples, making use of the standard rand() function. These formed the 10,000 pairs required to generate the 10,000 samples for the Gaussian approximation, using the above equation. The following graph shows the results of the test, revealing the familiar bell shape associated with a normal distribution.

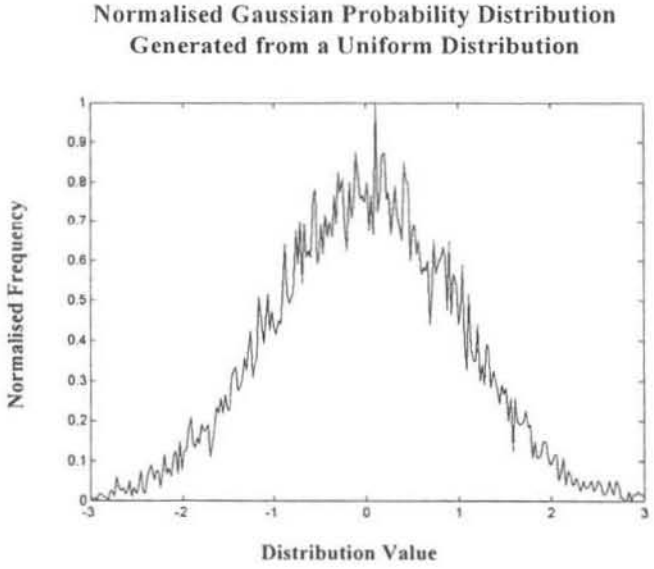

Figure 11.3 – Gaussian Deviate from Uniform Distribution

Once these samples from the Gaussian distribution have been generated, they can then be used in the next stage, which is the generation of the Rayleigh random variables

11.2.2 Generating Uncorrelated Rayleigh Random Variables

The generation of accurate uncorrelated Rayleigh Random variable samples requires the manipulation of 2 Gaussian random variable samples. Further, the desired mean of the distribution needs to be specified before calculation can proceed. The variance of the distribution is actually a function of the mean, and so specifying the mean automatically determines the variance. The equation, which is used in the generation of the Rayleigh samples, makes use of two normal distribution samples:

$$A_R = \sqrt{\left(N_1 \cdot \sqrt{\frac{2.M_R}{\pi}}\right)^2 + \left(N_2 \cdot \sqrt{\frac{2.M_R}{\pi}}\right)^2} \qquad \text{.....Equation 11.4}$$

- N_1 and N_2 are two Gaussian random variables
- M_R is the desired mean for the Rayleigh random variable
- A_R is the Rayleigh random variable, in this case, an Amplitude coefficient

Further

$$Var = \left(2 - \frac{\pi}{2}\right)\sigma_g{}^2 \qquad \text{.....Equation 11.5}$$

and

$$\sigma_g = \sqrt{\frac{2 \cdot M_R}{\pi}} \qquad \text{.....Equation 11.6}$$

In this generation technique, the standard deviation is a function of the desired mean. For simulation purposes, the mean should ideally be set to the path loss attenuation. However, for simulation purposes, the mean will be set to 1, indicating that, on average, the received signal will not be attenuated. Simulation of this generation technique produces the following graph:

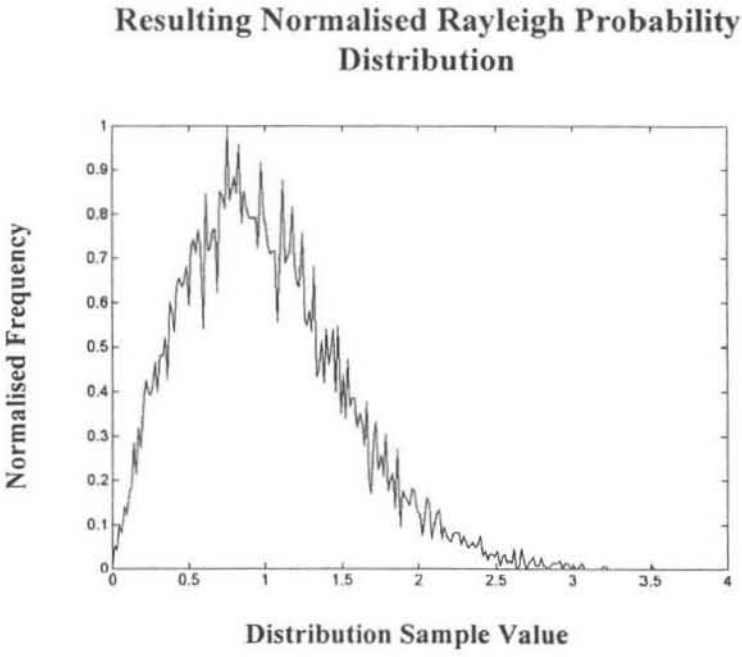

Figure 11.4 – Rayleigh Deviate from Gaussian Distribution

In Figure 11.4, the uncorrelated Rayleigh samples were taken from a space consisting of 10000 points, with a mean of 1. The measured mean of the given sample space is 1.0074, showing that the generator produces not only the correct Rayleigh distribution graph, but also the correct mean.

11.3 Implementing Uncorrelated Rayleigh Fading

To implement Rayleigh fading on the software radio system, the DSP card was used to generate the samples. This is because the generator suggested in the preceding pages, while very accurate, requires processor-intensive calculations. In addition, the DSP is incapable of producing a new sample from the random variable space on every byte that is transmitted. So, a third option was employed – that of generating samples and storing them in a long byte-valued array. Then, on each transmit interrupt, the Rayleigh sample is read from the array and transmitted to the software radio. Once the end of the array is reached, the DSP loops back to the beginning. This generates a satisfactory stream of Rayleigh samples. From the perspective of the software radio implementation, it is necessary to pack the stream of Rayleigh samples into 8-bit values, used to multiply the main transmitted signal. This requires that the values calculated in the DSP be correspondingly scaled. A scaling factor of 64 has been chosen to correspond with the maximum, realistic value attained in the distribution – that of 4 (4*64=256). In reality, this means that the simulated transmitted signal may be received with up to 4 times its original strength (due to multi-path effects) although, as seen from Figure 11.4, this is highly unlikely. Once the signal value has been multiplied with the Rayleigh coefficient, the effect of the scaling has to be removed. This is achieved in the software radio by a shifting the post-multiplication storage register to the right by 6 bits.

The rate of update is limited in this technique. The serial link from the DSP to the software radio is set at 2.048 Mb/s. This means that the byte transfer rate is 256kBytes/s. However, the transmitted signal is made up of 8 samples per symbol, with a symbol rate of 1.024MS/s. Thus, the Rayleigh fading coefficient will change every:

- 32 samples or
- 4 symbols

In this case, a symbol represents the same period as a chip, meaning that 1 bit will experience 4 different fading coefficients (since the processing gain is set at 16).

The Rayleigh-modified transmit signal can then be observed through the DACs on the software radio.

11.3.1 Rayleigh Fading Results

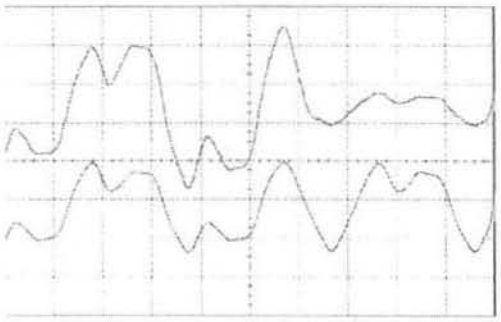

Figure 11.5 – Amplitude Amplification due to Rayleigh Fading

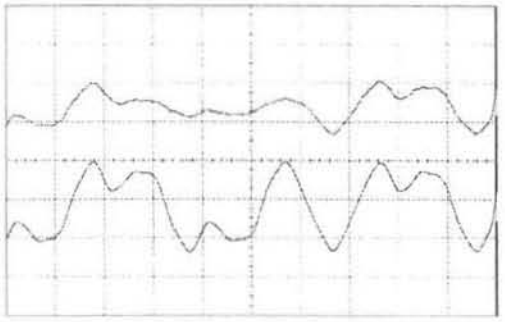

Figure 11.6 – Amplitude Attenuation due to Rayleigh Fading

Figure 11.5 and Figure 11.6 show the result of applying Rayleigh fading to a signal. In both graphs, the bottom waveform is the desired signal while the top waveform shows examples of the desired signal under the relevant Rayleigh fading conditions.

11.4 Implementing an Additive White Gaussian Noise Generator

The second main distortion that needs to be simulated is that of white noise. Here, it is possible to use the software radio for the generation of the noise. However, as is generally the case with an FPGA implementation, the requirement of joint speed and resource optimisation has to be considered in preference to accuracy. Obviously, an FPGA has no pre-defined random number generators, so these had to be designed before the AWGN could be produced. To establish a context, it must be noted that a uniform random number generator is used to produce the normal deviate. However, before this generator is fully specified, its mathematical importance needs to be displayed, and this can only be seen when the structure of the normal distribution generator is examined.

11.4.1 Specifying a Fast Normal Distribution Generator

The system under consideration generates samples at 8.192MS/s. Thus, to generate a new Gaussian variable on each new sample will require a generator running at 8.192MHz. Considering the mathematical complexity involved in generating the Gaussian samples for the Rayleigh fading component (Equation 11.3), this approach cannot realistically be ported to the software radio for the purposes of generating AWGN. Instead, the stringent requirement of accuracy needs to be relaxed and a viable alternate sought.

One particular solution is quite elegant, incorporating the ability to alter the mean and standard deviation. The flexibility inherent in being able to specify the standard deviation is essential, since it allows for the generation of AWGN with a predictable SNR with respect to a random input signal. This is because of the following simple statistical fact:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad \text{.....Equation 11.7}$$

- $\bar{x}$ is the mean of the set of numbers denoted by x_i
- n is the number of elements in the set
- S is the resulting standard deviation

For sufficiently large n, the standard deviation will approach the RMS value of the set of x. This is crucial in the final generation of noise with a specific Signal to Noise ratio, where the ratio is performed with RMS values.

Derivation of the generating function

[Nievergelt, Farrar & Reingold, 1974], [22], record a fast technique for generating a normal deviate, given a uniform random variable. Therefore, before a mathematical expression can be found for the fast normal distribution generator, the uniform random distribution needs to be closely examined. Following this will come its logical incorporation into the generation of the normal distribution.

First, a brief discussion of some elementary statistics will create a good basis for the discussion.

$$\Pr(a \le x \le b) = \int_a^b f(x)\,dx \qquad \text{.....Equation 11.8}$$

$$1 = \Pr(\alpha \le x \le \beta) = \int_{\alpha}^{\beta} f(x)\,dx$$ ..... **Equation 11.9**

where $[\alpha, \beta]$ is the interval over which f(x) is defined. The uniform distribution has a constant density function such that

$$1 = \Pr(0 \le x \le 1) = \int_{0}^{1} c\,dx = c$$ ..... **Equation 11.10**

Thus, it is easy to see that the probability density function for a uniform distribution is simply

$$f(x) = 1$$ ..... **Equation 11.11**

Two other important parameters of the uniform distribution are:

- The mean:

$$\mu = \int_{\alpha}^{\beta} x f(x) dx = \frac{1}{2}$$ ..... **Equation 11.12**

- The standard deviation

$$\sigma = \sqrt{\int_{\alpha}^{\beta} (x-\mu)^2 f(x) dx} = \sqrt{\frac{1}{12}}$$ ..... **Equation 11.13**

Approximating the normal (Gaussian) distribution:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x-\mu)^2/2\sigma^2}$$ ..... **Equation 11.14**

requires the use of the Central Limit Theorem. Here, if a set of independent random numbers is chosen from an arbitrary distribution f(x), which has a mean μ and a standard deviation σ, then there exists a distribution:

$$y = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i - \mu}{\sigma}\sqrt{n} \qquad \text{Equation 11.15}$$

which approaches a normal distribution having a mean of 0 and a standard deviation of 1. In the case of the uniform distribution

$$y = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{2}}{\sqrt{1/12}}\sqrt{n} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{2}\right)\sqrt{12n} \qquad \text{Equation 11.16}$$

To specify the mean M and standard deviation S for the above approximation, the following equation may be used:

$$y = M + S\left(\frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{2}\right)\sqrt{12n} \qquad \text{Equation 11.17}$$

This approximation has accuracy directly proportional to n. However, as will be seen, n can be as small as 4 for a sufficiently accurate and high-speed normal distribution generator. As a result, with as few as 4 samples from a uniformly distributed sample space, a further sample from the normal distribution may be obtained. Thus, the importance of the uniform random number generator has been established, and a suitable form needs to be found. Remembering that the uniform random number generator will reside in an FPGA, the two crucial requirements of minimal resources and maximum speed need to be used as criteria.

11.4.2 Uniform Number Generation

There are several forms of uniform number generators available. Not all are suitable for incorporation in the proposed normal distribution generator. What follows is a brief examination of available structures.

Linear Congruential Generators

The simplest uniform number generator is that offered in the ANSI C library. In their monumental work on the application of the C language, [W. H. Press, S. A. Teukolsky, W. T. Vetterling & B. P. Flannery, 1992], [21], record the generating equation for this simple uniform random number generator as being:

$$I_{j+1} = aI_j + c \pmod{m}$$ .....Equation 11.18

- m is the modulus
- a is a positive integer called the multiplier
- c is a positive integer called the increment

Obviously, this generator will repeat itself, and an upper limit of length m can be placed on the sequence. However, this generator has a further undesirable feature: sequential correlation on successive calls. Therefore, with a large number of calls to the generating function, a pseudo-random, uniform distribution may be generated. Therefore, while easily to implement, the accuracy (as tested in simulation) is unacceptable.

Generation of Random Bits

A generator that is more adapted to the hardware environment, producing considerably greater accuracy in terms of uniform distribution approximation, is a primitive polynomial generator. This polynomial consists of coefficients from the set {0,1}. It is represented either as a true polynomial equation, or as a list of those polynomial powers. For example:

$$x^{31} + x^3 + x^0$$ .....Equation 11.19

is a generating polynomial of order n = 31 that may be equivalently represented as:

$$(31,3,0)$$

This generating polynomial or list defines a means of deriving a new bit based on some or all of the previous n bits. In addition, if the generating polynomial is correctly designed, and if the starting seed is non-zero, then this generator is guaranteed to be of maximal length. This means that it will only repeat after 2^{n-1} iterations. There are several proposed polynomials for generating these maximal length sequences. However, before the performance is investigated, it must be pointed out that there are two methods of implementing this polynomial generator.

Method One

The following diagram helps in explaining the operation of the generator under method 1:

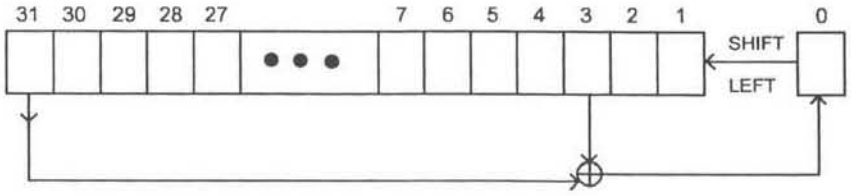

Figure 11.7 – Feedback Method One for Uniform Random Generator

For the given example,

$$b_0 = b_{31} \; XOR \; b_3 \qquad \text{.....Equation 11.20}$$

a new bit (b_0) is generated, the 31 bits are shifted one to the left (meaning that b_{31} is lost) and b_0 takes the place of b_1. This exclusive OR relationship is easily implemented in hardware. However, there is a further aspect to consider. [W. H. Press et al, 1992], [21], state (quite correctly) that sequential bits from this generator do not produce a uniform distribution over the range 2^n. Instead, all of the bits need to be replaced before the sequential numbers may be considered to be truly uniform in their distribution. In the case of the example, this means that 31 bits will need replacing before the next number has been generated. From an implementation perspective, a thorough investigation needs to be made of the effect of the length of the polynomial generator. The length plays two vital roles:

- It is inversely proportional to the resulting generator speed
- It is directly proportional to the accuracy of the generator

These two factors, obviously, play against each other and a suitable trade-off has to be found. In the case of method 1, the length of the generator plays a vital role, since the speed degradation is substantial in the case of longer generators and accuracy has to be limited. This is because on each clock edge, all n existing bits are modified. The implemented logic is fairly slow, as seen in Table 11.1.

Table 11.1 – Analysis of Polynoimial Order and Implementation Parameters for Method 1 Uniform Generator

Polynomial Order (n)		Maximum Clocking Speed (MHz)	Number of Logic Cells Used
20	(20,3,0)	47	33
21	(21,2,0)	37	35
22	(22,1,0)	18	39
23	(23,5,0)	63	36

24	(24,4,3,1,0)	17	65
25	(25,3,0)	37	41
26	(26,6,2,1,0)	15	77
27	(27,5,2,1,0)	15	79
28	(28,3,0)	37	44
29	(29,2,0)	27	47
31	(31,3,0)	37	48
32	(32,7,5,3,2,1,0)	9	N/A
33	(33,6,4,1,0)	13	N/A
35	(35,2,0)	23	56
39	(39,4,0)	37	60
63	(63,1,0)	8	94

Table 11.1 records the results for method one of the uniform generator. Since each generator needs to replace its register bits entirely before a new uniform number can be generated, the author made use of "for loops" when instantiating the entities in VHDL. Since the registers all differ in size according to column 1 of Table 11.1, the length of this "for loop" also varies. Further, the number of bits that need replacement on a single loop count also varies according to column 2 of Table 11.1. As a result, the synthesis appears poor with results that vary quite widely. Further, each Gaussian generator requires several uniform variables to generate a single normal variable, and the performance of this Gaussian generator will be correspondingly unpredictable. It is felt that this poor synthesis is as a result of the synthesis tool used, and not explicitly a fault with the VHDL code. As will be seen in method 2, however, by altering the contents of the "for loop", the performance can be improved and regulated.

Method Two

Method 2 incorporates "in place" replacements, followed by a shift. In other words, certain bits are modified, but rather than replacing the contents of another bit, the same bit is used to store the new value. The following diagram will help to explain this operation.

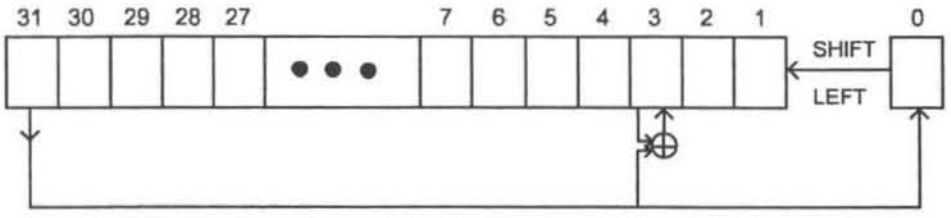

Figure 11.8 – Feedback Method Two for Uniform Random Generator

The procedure is as follows:

- The bits to be modified are individually XOR'd with the MSB
- The register is rotated to the left once

This procedure is repeated n times (n being the bit-width of the register), generating a new sample from the uniform distribution with range $\{0, 2^n\text{-}1\}$. In the example $(31,3,0)$

$$b_3 = b_3 \; XOR \; b_{31}$$
$$b_0 = b_{31}$$

.....Equation 11.21

This method not only produces acceptable distributions, with correct mean and standard deviations, but also results in a far smaller, quicker implementation, as seen in Table 11.2.

Table 11.2 – Analysis of Polynoimial Order and Implementation Parameters for Method 2 Uniform Generator

Polynomial Order (n)		Maximum Clocking Speed (MHz)	Number of Logic Cells Used
20	(20,3,0)	95	28
21	(21,2,0)	95	29
22	(22,1,0)	95	30
23	(23,5,0)	95	31
24	(24,4,3,1,0)	63	45
25	(25,3,0)	95	33
26	(26,6,2,1,0)	63	48
27	(27,5,2,1,0)	63	48
28	(28,3,0)	95	36
29	(29,2,0)	95	37
30	(30,6,4,1,0)	63	52
31	(31,3,0)	95	39
32	(32,7,5,3,2,1,0)	47	86
33	(33,6,4,1,0)	63	55
35	(35,2,0)	95	43
39	(39,4,0)	95	47
63	(63,1,0)	95	71

Here, the implementation frequency and logic cell usage is predictable. For single bit manipulation

- f (implementation frequency) = 95MHz

- lcc (logic cell count) = n+8 (n is the number of bits)

Comparison of Method 1 and Method 2

By way of an example, method 1 and method 2 were both used to generate a sample space of 80,000 points on a uniform distribution, used to produce 10,000 points on a normal distribution. Here, each method was used to construct 2 parallel uniform generators, each producing 4 samples in the uniform space per 1 sample in normal space. Both were set to have a mean of M=0, a standard deviation of S=2 and a register length n=3

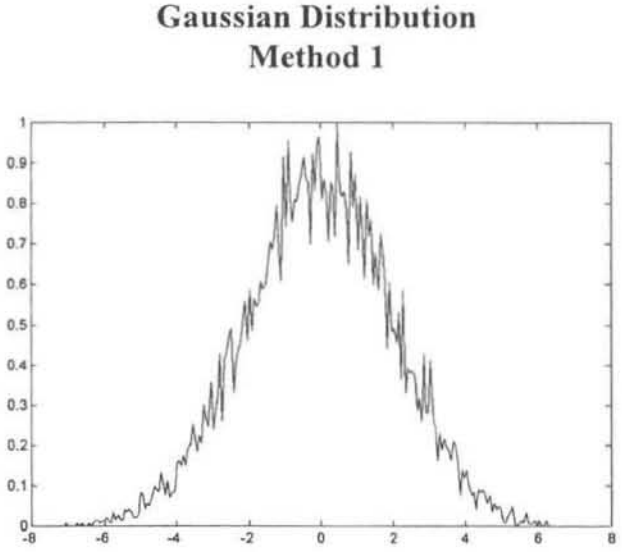

Figure 11.9 – Gaussian Deviate Using Method One Uniform Random Generator

Actual Mean = -0.0185

Actual Standard Deviation = 2.0048

Method 2

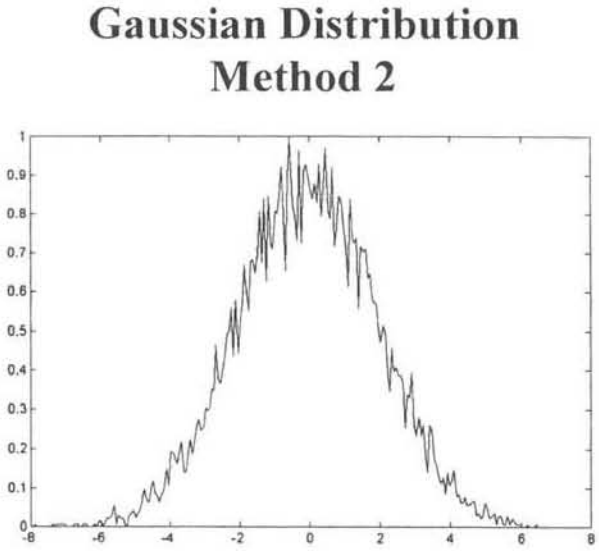

Figure 11.10 – Gaussian Deviate Using Method Two Uniform Random Generator

Actual Mean = -0.0364

Actual Standard Deviation = 1.9861

11.4.3 Implementation of Specific SNR

Using Method 2 as the preferred generating technique for the uniform sample space, it is possible to generate a Gaussian sample space with a fairly accurate mean and standard deviation, in addition to minimising FPGA resources while maximising the speed of the resulting entity. With these tools, it is possible to create an AWGN generator with noise at a specific RMS level. This is seen in the standard deviation of a large sequence of samples from a random variable space, of mean 0. The standard deviation is an excellent approximation of the RMS level of the samples. In addition, taking advantage of the Walsh code levels, it is possible to predict the RMS level of a random data signal encoded with Walsh codes. Armed with these two facts, the SNR can be defined as the ratio between the square of the RMS value of the signal and the square of the RMS value of the Gaussian samples:

$$SNR = 10\log_{10}\left(\frac{V^2}{\sigma^2}\right) \qquad \text{.....Equation 11.22}$$

- V = predicted RMS value of signal
- σ = standard deviation approximation of RMS value of the noise

It is possible to predict the RMS value of the input signal given that

- It is a binary signal with mean 0.5
- It is encoded with a NRZ Walsh code

With this knowledge, the following is true:

$$V = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{n} w_{ij}\right)^2} \qquad \text{.....Equation 11.23}$$

- N is the number of codes to be used in the system
- n the length of the code
- w_{ij} is bit j of Walsh code I

Since

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} x_i^{\,2}} \qquad \text{.....Equation 11.24}$$

and since the number of codes used in the system determines V, it is possible to fully specify the SNR by setting the value of σ:

$$SNR = 10\log_{10}\left(\frac{V^2}{\sigma^2}\right) \qquad \text{.....Equation 11.25}$$

$$\sigma = V.10^{-SNR/20}$$

Now, V can attain a maximum value of 4, while SNR can attain a minimum value of 0, setting σ_{max} at 4. Since the SNR can be nicely encoded in a byte, packaged and sent to the FPGA from the DSP, the following has been set:

- MSB = 1 to activate the noise generator
- Bits 6->0 are used to encode the σ value. This involves simply scaling σ by 32 to achieve a maximum value of 128.

Further, the noise is normalised in a sense that a signal stream of amplitude 1 will allow for the correct generation of noise of the desired SNR. The system is different in that the signal is comprised of a set of raised cosine pulses. It has been determined experimentally that the average value of the pulses under consideration is 76. The following should also be noted:

- Two uniform generators are used, each supplying 4 samples to make the number of uniform samples equal to 8.
- The uniform numbers are generated at 32.768 MHz, to allow for generation of Gaussian samples at 8.192 MHz
- x_i is scaled by a factor of 256
- $S_2 = S*32$

$$y = S\left(\frac{1}{n}\sum_{i=1}^{n} x_i - \frac{1}{2}\right)\sqrt{12n} \qquad \text{.....Equation 11.26}$$

$$= \frac{S\left(\frac{1}{8}\sum_{i=1}^{8} x_i - 128\right)9.79}{256}$$

$$\therefore y_2 = \frac{S_2\left(\frac{1}{8}\sum_{i=1}^{8} x_i - 128\right)9.79 \cdot 76}{256 \cdot 32}$$

$$\approx \frac{S_2\left(\frac{1}{8}\sum_{i=1}^{8} x_i - 128\right)768}{256 \cdot 32}$$

$$= \frac{S_2\left(\frac{1}{8}\sum_{i=1}^{8} x_i - 128\right)1024 \cdot 3}{256 \cdot 32 \cdot 4}$$

$$= \frac{S_2\left(\frac{1}{8}\sum_{i=1}^{8} x_i - 128\right)1024 \cdot 3}{32768}$$

Confusion may arise over the exact representation of this formula. It has been optimised for speed in the FPGA. To this end, the majority of the multiplications and divisions are performed as shift operations. So, to generate one Gaussian sample, there are 8 additions, one subtraction, one multiplication and one shift operation. Since the sum is multiplied by 1024 and divided by 32768, it is actually only divided by 32. This corresponds to the one shift operation that acts to the right by 5. The multiplication does not consume large amounts of resources since it involves a constant – the small value of 3.

11.4.4 Additive White Gaussian Noise Results

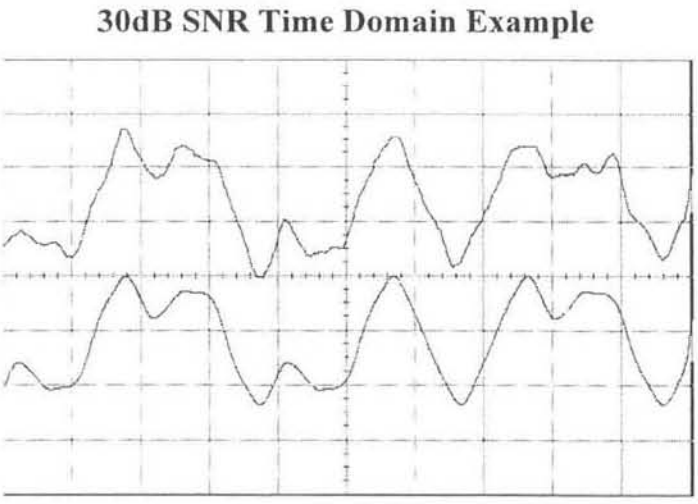

Figure 11.11 – Measured Time Domain Results for 30dB SNR

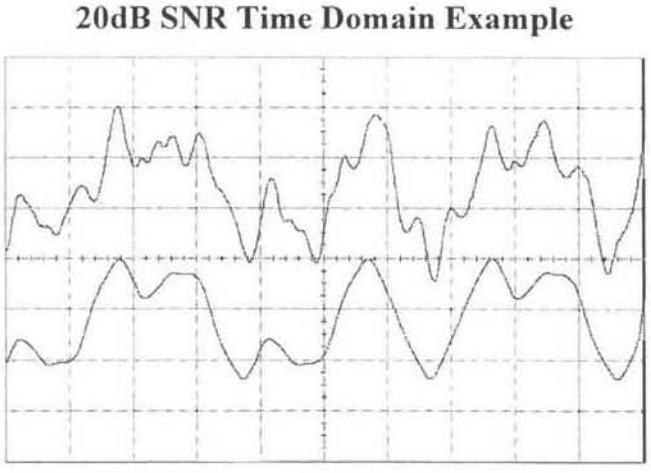

Figure 11.12 – Measured Time Domain Results for 20dB SNR

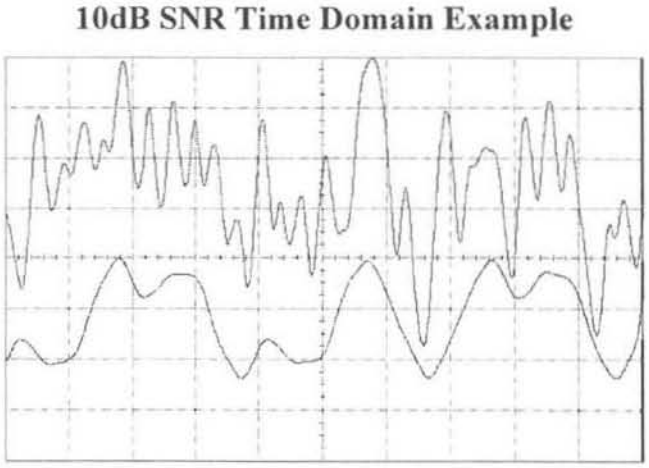

Figure 11.13 – Measured Time Domain Results for 10dB SNR

Figure 11.11 to Figure 11.13 show the time-domain effect of the Additive White Gaussian Noise generator. In each of the figures, the bottom trace is the multi-level CDMA signal without noise. This signal has already been subjected to the transmitter filter, producing the band-limited trace seen at the bottom of each figure. The same test signal has been used in each of the cases, with the frame sync acting as the synchronising signal. The AWGN generator, described in mathematical terms in this chapter, is then used to generate noise samples of a specific signal to noise ratio (relative to the transmitted signal). This is then added to the multi-level signal to produce the distorted trace seen at the top of each of Figure 11.11 to Figure 11.13. Since the frequency-domain results have already been presented in Figure 11.9 and Figure 11.10, these time-domain results aid in the testing of the generator for SNR generation. This was achieved by making use of the Agilent oscilloscope used for test measurements. Since the oscilloscope included mathematical features such as addition, division and signal RMS value, it was useful in validating the SNR levels produced. The oscilloscope was used to subtract the bottom trace from the top one for each of Figure 11.11 to Figure 11.13. The RMS value of the resultant trace was then divided by the RMS value of the bottom trace. On averaging the dB value of this ratio, it was indeed confirmed that the signal generator was producing the correct SNR in each case.

11.5 Personal Computer Simulation

To utilise the entities developed so far in this chapter, it is necessary to provide them with suitable data and control signals. The data is used in the simulation process to ascertain the Bit Error Rates, while the control signals are necessary to set the Signal to Noise Ratios for a particular simulation. There are three distinct processes involved in collecting the simulation parameters, processing them and then applying them in a simulated environment. The following diagram shows these layers, with the associated software development tools used to generate them:

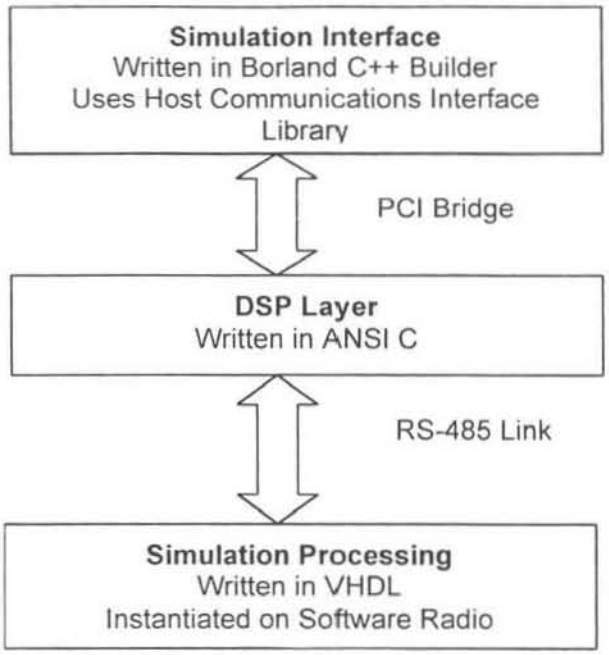

Figure 11.14 – Simulation Layers

11.5.1 Simulation Interface

The simulation interface serves several functions. Primarily, it is a parameter collection point and result depository. The parameters collected include the start SNR, end SNR, number of simulation points and the number of bits per simulation point. Further, it makes use of a bi-directional interfaces with the DSP layer in order to set parameters and read back counting variables. These counting variables count the number of bits received, along with the number that are erroneous. Making use of standard Windows timers and the HCIL library routines, the simulation interface polls the DSP card at regular intervals to retrieve the necessary data and update the SNR parameters.

The HCIL Interface Routines

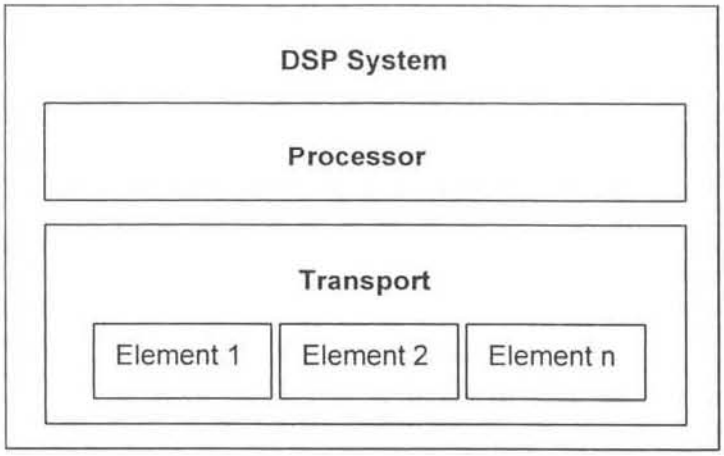

Figure 11.15 – The HCIL Hierarchy

The Host Communications Interface Library, provided by Blue Wave Systems, provides a flexible means of interfacing with the DSP system. The developer can choose to make use of either the C or C++ libraries. This project made use of the C++ routines, and Figure 11.15 reflects the inherent encapsulation associated with C++ objects. The HCIL defines a DSP System that encapsulates a processor interface and a transport interface. Once the programme for the DSP system has been loaded via the Processor object, the Transport object can be used to set up elemental transfers. Each element in the Transport object is associated with a specific memory location on the target DSP board, and bi-directional transfers are established between the PC and the DSP system using these transport elements. This allows the Simulation Interface to read or set the following parameters:

- Number of bits sent
- Number of erroneous bits
- Number of CDMA users
- The Signal to Noise Ratio parameter

From this, it is possible for the simulation interface to set the SNR to the next correct value once the correct number of bits has been sent. There is a minimum count resolution of 256000*T, where 256000 is the bit rate, and T is the period of the Windows timer. Obviously, with other interrupts occurring in the system, the count resolution may increase above this minimum value.

11.5.2 The DSP Layer

The DSP layer is responsible for receiving the data from both the PC and the Software radio, formatting it correctly and passing it on to the correct layer. From the PC, it will receive data

concerning the current SNR and the CDMA codes which are active in the system. It receives the current bit count and bit error count from the software radio, formats these for 32-bit memory transfers, and stores them in the correct location. A transport element, addressing this point from the PC, will retrieve this information. The DSP layer has been written in ANSI C.

11.5.3 The Simulation Processing

The software radio is responsible for processing the data and control signals to perform the simulation itself. The control signals inform the platform of the current SNR level, and the specific codes to use in the simulation. In turn, the software radio counts the number of bits that it processes, in conjunction with the erroneous bits. These are then packed in the following manner:

- The first 26 bits in a frame represent the bit count
- The second 26 bits in a frame represent the erroneous bits in the count

The DSP layer has the task of processing these numbers 8 bits at a time before delivering two 32-bit numbers to the simulation interface via the necessary transport elements. The software radio makes use of the RS-485 connection to report the count values to the DSP layer.

11.6 Results

This section presents observed system performance when the two channel distortion techniques shown in Figure 11.1 are activated. This corresponds to subjecting the channel to combinations of various AWGN and Rayleigh fading settings. 11.6.1 presents the results of the system where only AWGN affected the communications channel. The system performs well under these conditions since the filters and the MMSE detector have been designed to suppress the effects of noise. 11.6.2 presents the performance results of the system in the presence of Rayleigh fading and AWGN for a variety of signal to noise ratios. Here, however, the system does not perform as well, since Rayleigh fading is not compensated for.

11.6.1 Results of System Subjected to AWGN

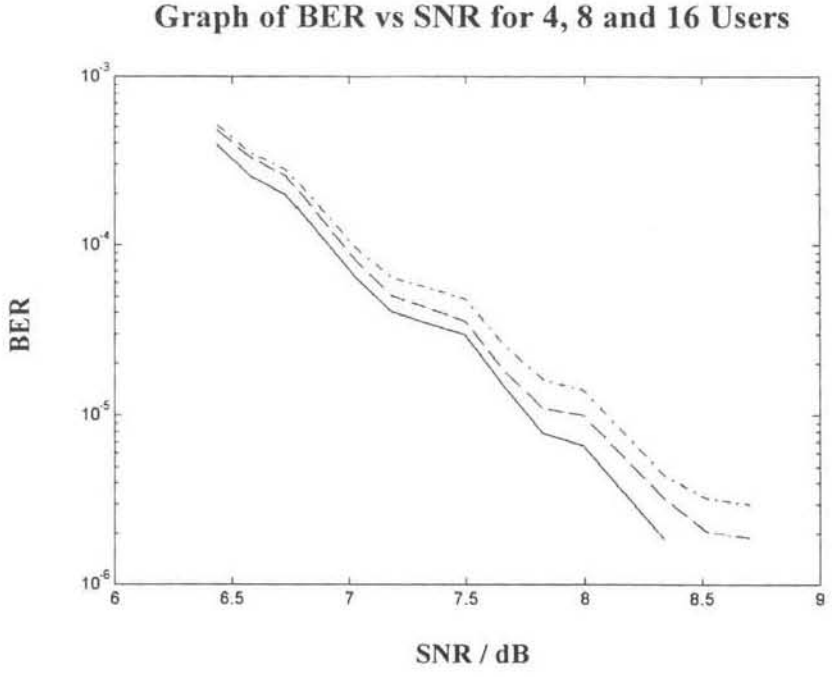

Figure 11.16 – Simulated BER vs SNR

Figure 11.16 shows the results obtained when simulating on the software radio. The graph is not particularly smooth, but it does show the general trend associated with decrease in BER as SNR improves. A point of interest is the very low MAI generated when using Walsh codes. The top line in the figure represents 16 users, and there is very little improvement obtained in dropping this to only 4 users (the bottom line). The "curve" has this stepped appearance because of the low resolution of the control signal for the SNR value. This, being only 8-bits, results in slight inaccuracies that do not average out over the run of the simulation, but rather have an accumulative effect. This situation could be improved by using a higher resolution for the encoding of the SNR value. However, the great speed advantage of this simulation technique makes it appealing to investigate and improve its performance further. This particular platform is capable of simulating 1.536 million bits in a minute, a performance far higher than most simulations running on a high-speed personal computer.

11.6.2 Results of System Subjected to Uncorrelated Rayleigh Fading

The next stage in the simulation process involves gathering results when the system is subjected to Rayleigh fading. This corresponds to activating the Rayleigh fading element seen between the transmitter and receiver filters depicted in Figure 11.1. Of paramount importance in this operation is the speed at which the transmitted information can be subjected to the Rayleigh fading coefficients, since this will determine the resolution of the depth of the fades. As has

been reported in 11.3, this maximum rate corresponds to transmitting a different byte-encoded Rayleigh coefficient in each of the time slots on the SCBUS. This 256kHz update rate relates to a maximum fading rate of 4 chips. This means that the Rayleigh coefficient will affect only 4 chips before a new value is loaded. As the speed of this fade increases, so the system will be able to average its effect and thereby reduce the associated BER. However, as the fade depth increases, so the system will suffer from degraded performance. This is seen in the results recorded in Figure 11.17 to Figure 11.19. For these results, the Rayleigh fading mean was set to 1. As seen in Equation 11.5 and Equation 11.6, the variance is mathematically derived from the specified mean. Since the mean was set to 1, the variance can be calculated to be 0.2732. For the actual generation of the Rayleigh fading, the variance was found (due to rounding errors) to be 0.2788.

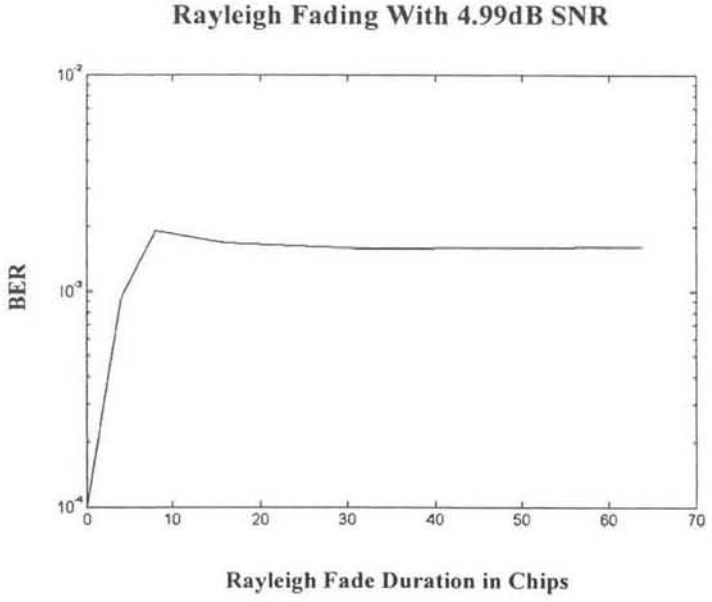

Figure 11.17 – Errors introduced by Rayleigh Fading at 4.99dB SNR

Figure 11.17 records the results for a system of 4 users when the channel is subjected to Rayleigh fading in addition to additive white Gaussian noise. As seen in the figure, the noise was set to create a 4.99dB SNR and this remained constant during the simulation. However, the Rayleigh fade duration was increased from 4 to 64 chips and the results were observed. It was noted that increasing the fade duration above 20 chips had very little effect on the BER, which, as seen in Figure 11.17, remained fairly constant in this portion of the graph. However, the graph peaks at a fade depth of exactly 8 chips - corresponding to half the length of the Walsh codes used. Rayleigh fading was turned off when the fade duration was set to 0 chips, corresponding to a simple AWGN channel.

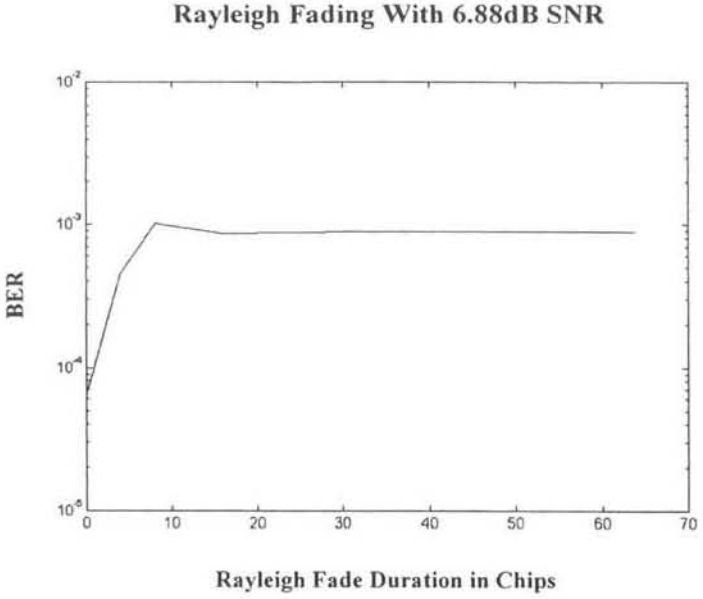

Figure 11.18 - Errors introduced by Rayleigh Fading at 6.88dB SNR

Figure 11.18 also records the results for a system of 4 users when subjected to both Rayleigh fading and AWGN. Here, however, the AWGN level was set to produce a constant 6.99dB SNR during the simulation. As in Figure 11.17, the BER peaks when the fade duration is set to exactly half the length of the Walsh codes that were used to encode the users' data streams. Once again, the graph displays a constant value for fade depths above 20 chips, and Rayleigh fading was turned off when the fade duration was set to 0 chips.

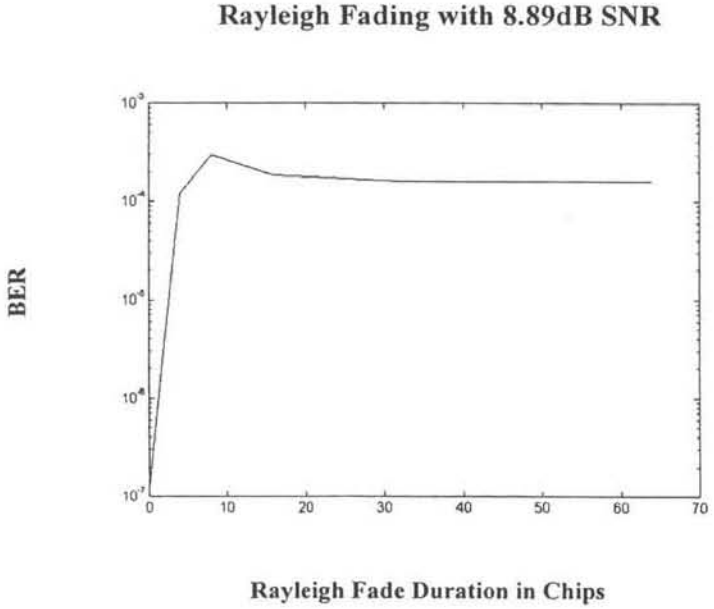

Figure 11.19 - Errors introduced by Rayleigh Fading at 8.89dB SNR

Figure 11.19 shows the final set of results for a 4-user system where the channel is subjected to both Rayleigh fading and AWGN. The graph displays the same characteristic for fade durations above 20 chips where the BER remains fairly constant. Further, the peak BER was observed at a fade duration of 8 chips.

The system, as it has been designed, does not cater for Rayleigh fading. However, as seen in the filtering techniques, and the nature of the MMSE filter, the system has been designed to cater for AWGN. The results in 11.6.1 show that the system performs well in AWGN channels. However, since no channel estimation has been built into the system as it stands, it has no means of cancelling or predicting the effects of Rayleigh fading. As a result, the system performs poorly when subjected to both AWGN and Rayleigh fading.

11.7 Conclusion

This chapter has revealed, once again, that the software radio embodies great flexibility in design and implementation. This flexibility has been used to simulate a simple communications channel, incorporating Additive White Gaussian Noise, and Uncorrelated Rayleigh fading. This process involves many interfaces and techniques for passing data and control parameters between these interfaces. However, without the underlying statistical foundation, the system could not have been developed. The software radio also proves to be a very useful tool in quickly and easily transforming mathematical processes into hardware implementations. This is due largely to the abstraction of the developer from the hardware issues. Finally, this chapter has shown that moving the computational burden of a simulated environment from a personal computer to a dedicated platform has a great speed advantage.

Chapter 12 Conclusion and Recommendations

Wireless communications, especially in the cellular context, will continue to extend its influence on a global scale. As more users become informed and begin to explore the limits of new technologies, so the demand for improved performance will begin to draw on mathematical and technological innovations. This has already been the case, with new mobile communication standards emerging to meet the needs of a highly mobile population. However, this has not only placed pressure on engineers to develop powerful communications systems, but also to seamlessly incorporate the existing basis from which the success of cellular communications has been launched. This thesis has examined two related phenomenon – the rise of CDMA as a viable wireless communications technique, and the development of the software radio to deliver key flexibility options. CDMA has been shown to be a spread spectrum communications technique, enabling it to coexist with other narrow-band transmissions. Further, the software radio is one means of allowing a single device to be able to access services offered by these various communications systems. This feature is derived from its flexible, re-programmable nature, a characteristic that, as this thesis has shown, enables the device to be far more than just a communications tool.

While GSM presently dominates cellular communications as the wireless protocol of choice, it is envisaged that, with time, the battery of positive characteristics of CDMA will force it further into the limelight. This thesis shows that, while it has had several technological barriers in the past, the protocol offers many advantages over current techniques and is gathering momentum as mobile hardware and technology improves. Further, mathematical progress in the realm of multi-user detection techniques has seen improved performance and extended possibilities for CDMA. These techniques make use of, and detect, all the users on a CDMA communications channel simultaneously. This allows for performance gains over traditional correlating detectors, which treat other users as AWGN, thus precluding the use of the information contained in the structured nature of the CDMA signal. The advantages do require copious amounts of processing power, and the software radio is one platform that offers many of the features necessary to implement such a CDMA system.

The software radio is a radio where many of the functions can be implemented in a re-configurable processing block. Many of the modules described in this thesis have been implemented on the Flexible Radio Platform, the software radio offered by Alcatel Altech Telecomms. Since FPGA's form the backbone of this software radio offering, VHDL was used to programme the devices. This digital description language is ideally suited to the high-speed FPGA environment, and proved to be a capable tool in the development of the core CDMA routines. The design methodology, however, was tedious since, to accurately test a system, it had to be compiled through to completion. This ensured that the system performed adequately,

in addition to fitting in the FPGA. Newer simulators offer EDIF-level simulation of VHDL systems and thus integrate more tightly with the system design. However, these were unavailable for the implementation of this project. Using VHDL to generate the entities, and schematics to create the top-level interfaces, the CDMA system was successfully implemented on the software radio.

This thesis has shown that the software radio used in this project is a highly flexible and powerful tool for the development of mobile communications systems. It has also shown that the FPGA can incorporate many functions previously relegated to sequential Digital Signal Processors. Base-band shaping of the CDMA signal was undertaken in a set of matched Finite Impulse Response filters. In this implementation phase, it becomes imperative that the developer balance resource usage and speed in the FPGA-based system. As was shown, these two parameters are permanently at loggerheads and the developer needs to carefully balance the design so as to optimise both. With this in mind, many realisation structures were presented as possibilities for the FIR filters. This culminated in a promising new technique known as distributed arithmetic. This solution promises to produce FIR filters capable of running at the system clock speed.

Not only does the software radio perform well in a mobile communications environment, but it also extends the developer's tools to incorporate testing and debugging techniques. It was shown that, once the system has been programmed, that it may be subject to channel distortions to test its viability. This requires a few statistical generation entities and higher-level layers to control the distortion parameters. These were incorporated in a simulation environment in which the personal computer set and controlled the necessary parameters, while recording the associated bit error rates. It was seen that, not only could the developer test the system, but that the simulations could also be performed in real time. This offers a speed advantage over traditional personal computer aided simulations.

In spite of the success of the project, there are many recommendations that would allow the flexible radio platform to become an even more useful tool. First, multi-user detectors require the calculation of a matrix inversion that, especially in the presence of a rapidly changing channel, is computationally burdensome. As the thesis examined the software radio, so it became apparent that it is possible to use resource partitioning to overcome this limitation. Here, a DSP is used to calculate offline information, such as a matrix inverse. However, in this system, the link between the FPGA core and the DSP subsystem is a fairly slow RS-485 communication line. As a result, off-line calculations and updates are possible, but not at the speeds required by normal cellular communications. It is therefore a strong recommendation that, while this route is explored in future research implementation work, that another alternate be examined. This may include the use of the inter-FPGA communication bus for transferring signals to an off-board DSP chip.

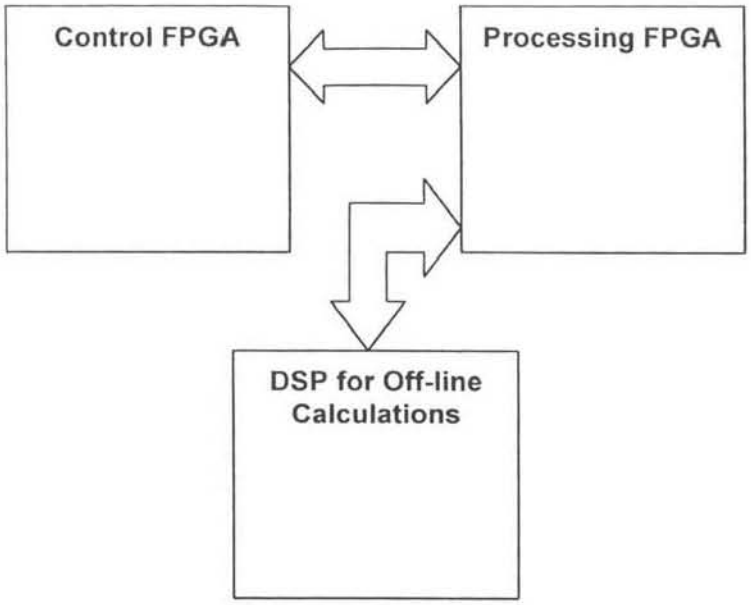

Figure 12.1 – Splitting the FPGA bus to transfer signals to an off-board DSP

Figure 12.1 shows how the off-line processing may be achieved in the DSP. Under these conditions, the processing FPGA can be used to package the data appropriately for serial transmission, while the DSP will have the task of channel estimation and matrix inversion. This resource partitioning would circumvent the speed and resource limitations encountered while implementing this project. Second, a further resource partitioning should be present in the form of a large, high-speed memory bank. When the partial summation techniques were investigated for base-band pulse shaping, it was seen that the FPGA technology used has very limited internal RAM. Further, the external RAM is also very limited. As a result, partial summations cannot be easily stored, especially in the case of high bit accuracy where the number of coefficients escalates exponentially. Nevertheless, the platform did offer most of the functionality expected of a software radio.

This thesis has examined two emerging technologies that serve to complement each other as much as they each stand as individual milestones. CDMA promises to boost the functionality and performance of cellular communications, while the software radio is seen as the tool for easily testing and deploying new communications protocols. Further, due to its re-configurable nature, the software radio also promises to seamlessly bridge the divide between the multitudes of communications protocols presently influencing mobile communications. Together, these technologies will influence tomorrow's world of mobile communications as they set new standards and reach for new limits in global cellular communications.

Chapter 13 References

[1] R. Prasad, "Overview of CDMA Evolution toward Wideband CDMA", *IEEE*, 1998.

[2] S. Verdu, "Minimum Probability of Error for Asynchronous Gaussian Multiple-Access Channels", *IEEE Transaction on Information Theory*, IT-32, No. 1, pp. 85-96, January 1986.

[3] J. Mitola III, "Software Radio Technology Challenges and Opportunities", *First European Workshop on Software Radios Keynote Address.*

[4] The Science and Technology Policy Program (SRI International-Washington, DC), "The Role of NSF's Support of Engineering in Enabling Technological Innovation - Phase II", Chapter 4: THE CELLULAR TELEPHONE.

[5] R. L. Pickholtz, L. B. Milstein, and D. L. Schilling, "Theory of Spread-Spectrum Communications – a Tutorial", *IEEE Transactions on Communications*, Volume COM-30, No. 5, pp. 855-882, May 1982.

[6] S. Glisic and B. Vucetic, "Spread Spectrum CDMA Systems for Wireless Communications". Artech House Publishers, 1997.

[7] Taub & Schilling, "Principles of Communications Systems". Second Edition, McGraw-Hill International, 1986.

[8] http://www.xsilogy.com/support/codes.html.

[9] R.E. Ziemer and R.L. Peterson, "Digital Communications and Spread Spectrum Systems".

[10] A. Grant and C. Schlegl, "Iterative Implementations for Linear Multi-User Detectors", February 3, 1999.

[11] Fraleigh & Beuregard, "Linear Algebra, 3rd Edition". Addison Wesley, 1995.

[12] Dr. J. H. Reed & Dr. B. D. Woerner, "Industry Trends Leading to the Development of the Software Radio Concept". http://www.softradios.com/trends.html.

[13] I. Seskar and N. B. Mandayam, "Software-Defined Radio Architectures for Interference Cancellation in DS-CDMA Systems", *IEEE Personal Communications*, Volume 6, Number 4, August 1999.

[14] M. Kruger, "User Manual For The I/O Processing Board Of The Dsp-Based Digital Transceiver System". Alcatel Altech Telecomms, 2000.

[15] Blue Wave Systems, "PCI/C6200 Technical Reference Manual". Version 0.90, June 1998.

[16] J. D. Gibson, "Principles of Digital and Analogue Communications, Second Edition". Macmillan Publishing Company, 1993.

[17] A.P. Clark, "Principles of Digital Data Transmission". Pentech Press, 1983.

[18] E. C. Ifeachor and B. W. Jervis, "Digital Signal Processing – A Practical Approach". Addison-Wesley Publishers Limited, 1993.

[19] ALTERA, "Implementing FIR Filters in FLEX Devices". February 1998, ver. 1.01.

[20] A. Ralston, "Mathematical Methods for Digital Computers". John Wiley & Sons, 1967.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical Recipes in C – The Art of Scientific Computing". Second Edition, Cambridge University Press, 1992.

[22] Nievergelt, Farrar, and Reingold, "Computer Approaches to Mathematical Problems". Prentice-Hall, 1974.

[23] Synopsys, "FPGA Express, VHLD Reference Manual". Synopsys, 1997.

[24] Alcatel Altech Telecomms, "Software Radio Main I/O Board Schematics & Assembly Drawings". AAT, 1999.

[25] N.S. Alagha and P. Kabal, "Generalised Raised-Cosine Filters", *IEEE Transaction on Communications*, Volume 47, No. 7, pp. 989-997, July 1999.

[26] James Miller, "Practical Communications Theory – Part 2". Spread Spectrum Scene, 1998: www.sss-mag.com/G3RUH/index1.html

[27] K.G. Beauchamp, "Walsh Functions and their Applications". Academic Press, 1975.

[28] H. Saarnisaari, "Robustness of the MUSIC algorithm to Errors in Estimating the Dimensions of the Subspaces: Delay Estimation in DS/SS in the Presence of Interference", University of Oulu, Telecommunication Laboratory, 1999.

[29] E.G. Strom, S. Parkvall, S.L. Miller, B.E. Ottersten, "Propagation Delay Estimation in Asynchronous Direct-Sequence Code-Division Multiple Access Systems", University of Florida/Royal Institute of Technology, 1994.

[30] T. Ostman and B. Ottersten, "Systems with Bandlimited Pulse Shapes", Royal Institue of Technology.

Chapter 14 Appendix

A.1 Kronecker Products

Consider the two matrices A (nxm) and B (pxo). Then, C=A*B, where * represents the direct product of A and B, will see C having the following properties:

- C will have n x p rows
- C will have m x o columns
- C is effectively the matrix B, with each element b_{ij} replaced with the matrix b_{ij} . A

$$A=\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad B=\begin{bmatrix} e & f \\ g & h \end{bmatrix}$$ ..... **Equation 14.1**

$$A \otimes B=\begin{bmatrix} e.a & e.b & f.a & f.b \\ e.c & e.d & f.c & f.d \\ g.a & g.b & h.a & h.b \\ g.c & g.d & h.c & h.d \end{bmatrix}$$

For example:

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix}$$ ..... **Equation 14.2**

Therefore, with Hadamard matrices, the order doubles with each application of the direct product. This has the obvious disadvantage of low flexibility, since if a system needs to cater for 129 users, PN sequences of length 256 will have to be used.

A.2 Iterative Matrix Inversion

Number of Operations in Gauss-Jordan Reduction

As a point of reference for gauging the success of iterative methods, the Gauss-Jordan matrix reduction scheme will be used. In this well-know technique, the (nxn) matrix to be inverted is augmented with the (nxn) identity matrix. After Gauss reduction of the matrix to be inverted,

with the same operations being mirrored on the identity matrix, the augmented matrix will contain the inverse of the initial matrix:

$$[A \mid I] \xrightarrow{Gauss-Jordan} [I \mid A^{-1}] \qquad \text{.....Equation 14.3}$$

For each of the n pivot points in the matrix, there is the following number of calculations:

- (n – 1) divisions
- (2n – 1) subtractions
- (2n – 1) multiplications

To complete the operation, there are n^2 normalising divisions, followed by the vector multiplication, which incorporates:

- (n^2) multiplications
- n(n – 1) additions

Grouping the divisions with the multiplications, and the subtractions with the additions, reveals the following total number of operations:

$$N_{mul} = 5n^2 - 2n \qquad \text{.....Equation 14.4}$$
$$N_{add} = 3n^2 - 2n$$

Number of Operations is a Jacobi Iteration

$$x_{i+1} = S^{-1}(Tx_i + b) \qquad \text{.....Equation 14.5}$$

Equation 14.5 shows the standard iteration step for all of the processes presented. It involves the multiplication of a matrix with a vector (Tx), the addition of two vectors (Tx + b) and a final multiplication of a matrix with a vector (S^{-1}(Tx+b)). From this, it is possible to derive the number of mathematical operations performed in a single iteration step for each of the iteration techniques. The ratio of the values in Equation 14.4 with those calculated for the iteration procedures will give an indication of the efficiency of the iteration technique. First, this section will examine the Jacobi iteration

Since T has a zero diagonal, and otherwise non-zero elements for Jacobi, Tx employs the following operations:

$$N_{mul} = n(n-1)$$
$$N_{add} = n$$ ..... **Equation 14.6**

Further, there are n additions in the vector addition step, while the final diagonal matrix multiplication with the vector has n multiplications. Therefore, the total operations for the Jacobi iteration is:

$$Nmul = n^2$$
$$N_{add} = 2n$$ ..... **Equation 14.7**

Number of Operations in a Gauss-Seidel Iteration

For any given algebraic sequence

$$(a, a+d, a+2d, ..., a+(N-1).d)$$ ..... **Equation 14.8**
$$\therefore S_N = N\left[a + \frac{d}{2}(N-1)\right]$$

From Equation 14.8, it is possible to derive the number of multiplications and additions necessary to carry out the Gauss-Seidel iterative matrix inversion procedure. Given Equation 3.7, we see that S is a lower and diagonal (nxn) matrix and T is a strictly upper triangular (nxn) matrix. The calculation of interest is:

In Equation 14.5, the inverse of S is also a lower and diagonal (nxn) matrix, and it is of interest to see how many additions and multiplications will be performed per iteration.

When a strictly upper triangular (nxn) matrix is multiplied with a (nx1) column vector, the following is true:

$$N_{mul} = \frac{n}{2}(n-1)$$
$$N_{add} = \frac{(n-1)}{2}(n-2)$$ ..... **Equation 14.9**

When a strictly upper triangular (nxn) matrix is multiplied with a (nx1) column vector, the following is true:

$$N_{mul} = \frac{n}{2}(n+1)$$
$$N_{add} = \frac{n}{2}(n-1)$$

.....**Equation 14.10**

There are a further n additions when Tx is added to b. As a result, in any iteration, the following number of operations occurs:

$$N_{mul} = n^2$$
$$N_{add} = n^2 - n + 1$$

.....**Equation 14.11**

Multiplication Iterative Efficiency

From the equations in this appendix on matrix inversion, it is clear that there is a point beyond which performing more iterations becomes more expensive than following direct techniques. The following graphs indicate the iteration limits:

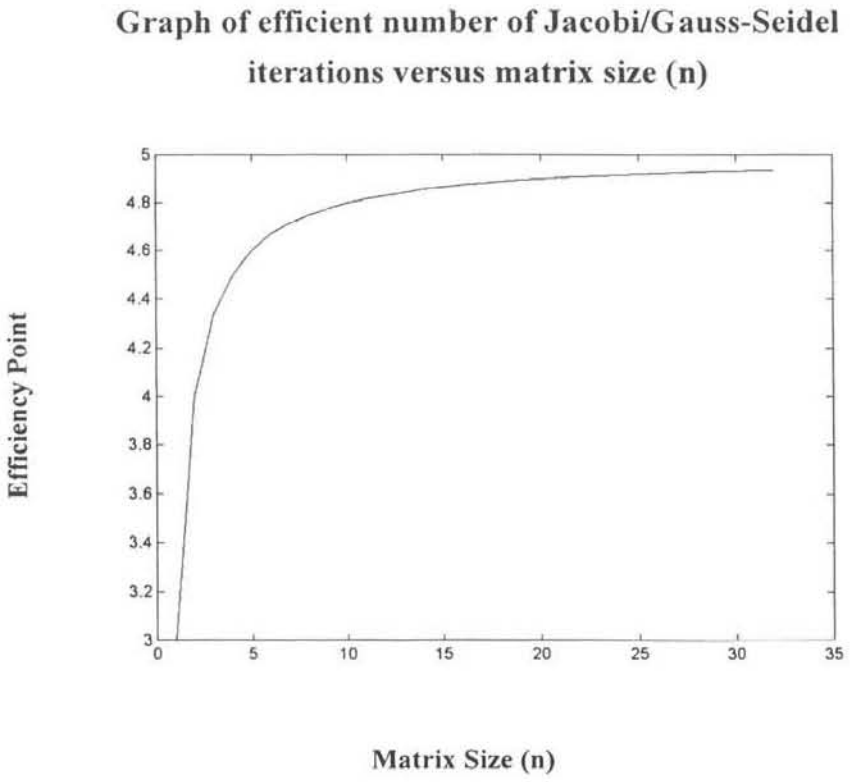

Figure 14.1 – Iteration Efficiency Plot for Jacobi/Gauss Seidel Iterative Techniques

Figure 14.1 shows the relationship between the matrix size and the efficient number of iterations. The efficient number of iterations is the point beyond which it is more efficient to use the Gauss-Jordan reduction technique, in preference to the Gauss-Seidel iterative scheme.

A.3 Gold Code Sequence Generation

Section 2.2.2 revealed the many desirable properties of Gold Codes. However, there are a limited number of preferred pairs that are suitable for the generation of Gold Codes. Table 14.1 summaries some of them.

Table 14.1 – Some Gold Code Preferred Pairs

Gold Code Length	Polynomial 1 (Hex)	Polynomial 2 (Hex)
31	25	3D
63	43	67
127	89	8F
511	211	259
1023	409	50D
2047	805	925