DAB implementation in SDR

Petro Pesha Ernest

Thesis presented in partial fulfilment of the requirements for the degree of

Master of Science in Electronic Engineering

at the University of Stellenbosch

Supervisor: **Prof. J.G. Lourens**

December 2005

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, except where indicated. It has not been previously submitted to any university for a degree in its entirety or in part.

SIGNATURE DATE

Abstract

The aim of this thesis is to implement a Digital Audio Broadcasting (DAB) system in a Software Defined Radio (SDR). The physical modulation part of the DAB transmitter for one of the transmission modes as well as its receiver is to be implemented and tested in the SDR. DAB transmission mode II is implemented.

A simulation is done first, which is followed by a real-time implementation in the SDR architecture. The simulation is implemented using the Microsoft Windows XP operating system and MATLAB. The real-time implementation of the system is done under the Linux operating system, using XML and C++.

In the real-time implementation, one computer is used for both transmission and reception. Base-band transmission is used. The software implementing the transmitter generates the base-band signal and passes it to the Data Acquisition card (DAQ) installed in the computer. The software implementing the receiver, receives the signal from the DAQ and performs demodulation. The DAQ card performs both digital-to-analogue and analogue-to-digital conversions.

The results obtained showed that the implemented system works well. The theoretically predicted performance and practical performance agree remarkably well.

Opsomming

Die doel van hierdie tesis is om ‘n Digital Audio Brodicasting (DAB)stelsel te implementeer in ‘n Sagteware Gedefinieërde Radio (SGR). Die fisiese modulasie komponent van die DAB sender, sowel as sy ontvanger is in SGR geïmplementeer en getoets vir een van die transmissie modusse. DAB transmissie modus II is geïmplementeer.

‘n Simulasie is gedoen, gevolg deur ‘n intydse implementasie in die SGR argitektuur. Die simulasie het van die Microsoft Windows XP bedryfstelsel asook MATLAB gebruik gemaak. Die intydse stelsel het gebruik gemaak van die Linux bedryfstelsel en die programmeringstale XML en C++.

Tydens die intydse implementering word een rekenaar gebruik vir beide transmissie en ontvangs. Slegs basisband transmissie word gebruik. Die sagteware wat die sender implementeer, genereer die basisband sein en stuur dit vir die versyferingskaart (DAQ), wat in die rekenaar geinstalleer is. Die sagteware wat die ontvanger implementeer, ontvang die sein vanaf die DAQ en doen die nodige demodulasie. Digitaal-na-analoog en analog-na-ditaal omsetting word albei behartig deur die DAQ kaart.

Die resultate toon dat die geïmplementeerde stelsel goed werk. Die teoreties voorspelde resultate stem baie goed ooreen met die praktiese gemete resultate.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof. Johan G. Lourens for his time and effort during the development of this thesis, thank you for your guidance, patience and encouragement. I also want to thank Dr. G-J van Rooyen for his advice during real time implementation in SDR.

Special thanks go to the Dar es Salaam Institute of Technology (DIT) for financial support.

Additionally, I would like to thank my fellow SDR research group members and DSP lab members at the University of Stellenbosch for their friendship. Finally I would like to thank David Mwakyusa and Elias Mathaniya for their friendship and encouragement.

Contents

List of figures

List of tables

Glossary of abbreviations

ADC	Analog-to-Digital Converter
AM	Amplitude Modulation
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
COFDM	Coded Orthogonal Frequency Division Multiplex
DAB	Digital Audio Broadcasting (Eureka-147)
DAC	Digital-to-Analog Converter
DAQ	Data Acquisition card
DFT	Discrete Fourier Transform
DQPSK	Differential Quadrature Phase Shift Keying
EBU	European Broadcasting Union
ETS	European Telecommunication Standard
ETSI	European Telecommunication Standard Institute
FDM	Frequency Division Multiplexing
FFT	Fast Fourier Transform
FIB	Fast Information Block
FIC	Fast Information Channel
FM	Frequency Modulation
GHz	Giga Hertz
IBOC	In-Band On-Channel
ICI	Inter-Carrier Interference
ISDB-T	Terrestrial Integrated Services Digital Broadcasting
ISI	Inter-Symbol Interference
kbits/s	Kilobits per second
kHz	Kilo Hertz
Mbits/s	Megabits per second
MPEG	Moving Pictures Expert Group
MSC	Main Service Channel
OFDM	Orthogonal Frequency Division Multiplex
PCM	Pulse Coded Modulation

QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
SFN	Single-Frequency Network
SNR	Signal-to-Noise Ratio
UHF	Ultra High Frequency
VHF	Very High Frequency

Chapter 1

INTRODUCTION

1.1 Software defined radio

In recent years there has been an enormous proliferation of standards in broadcast (radio and television), in mobile, and in personal communications. Examples with high profiles currently include digital radio (DAB, IBOC), digital television, wireless LAN and mobile communications. These standards form the basis for an ever-growing number of sophisticated consumer electronic devices, each with the potential to sell in very high volumes. In typical designs, these complex standards are implemented using dedicated architectures, which are optimised to reduce chip costs to the absolute minimum. This approach to chip design leads to long product development times [31], with a high risk of problems being found late in the development cycle. Products developed using dedicated architectures are often difficult to upgrade in order to support changes to the standards, or to add new features [32][33].

Software Defined Radio (SDR) is one way to address these issues [31] [32][33]. By using a sufficiently powerful programmable architecture, many different transmission standards can be supported on a common platform. A radio system implemented on a programmable architecture can be upgraded in the field to fix bugs or to add functionality, and it can support new standards as they are defined, assuming that there is sufficient flexibility in the architecture.

Software Defined Radio (SDR) refers to the technology wherein software modules running on a generic hardware platform consisting of DSPs and general purpose microprocessors are used to implement radio functions such as generation of the transmitted signal (modulation) at the transmitter and tuning/detection of received radio signal (demodulation) at the receiver. In SDR, radio functions are performed by software. In this way, the radio functions traditionally defined by hardware components can in future be defined by software components in SDR. This feature makes the SDR operate on different frequency bands, standards and applications and makes it reconfigurable.

SDR technology facilitates implementation of some functional modules in radio system such as modulation/demodulation, signal generation, coding and link-layer protocols in software. This helps in building reconfigurable software radio systems where dynamic selection of parameters for each of the above-mentioned functional modules is possible. A complete hardware based radio system has limited utility since parameters for each of the functional modules are fixed. A radio system built using SDR technology extends utility of the system for a wide range of applications that use different link-layer protocols and modulation/demodulation techniques.

SDR technology can be used to implement military, commercial and civilian radio applications.

1.2 Thesis objective

The objective of this thesis is to implement a Digital Audio Broadcasting (DAB) system in the SDR, where the physical modulation part of the DAB transmitter for one of the transmission modes and its receiver is to be implemented in the SDR, tested and included in the SDR library.

1.3 Thesis layout

The layout of the remainder of this thesis is as follows:

Chapter 2: This chapter describes the theoretical background of the DAB system and practical considerations regarding its implementation (i.e. transmission standards and transmission modes).

Chapter 3: In this chapter the simulation of the physical modulation part of the DAB system is implemented. The DAB transmitter for transmission mode II and one of the receivers are simulated.

Chapter 4: This chapter describes the real time implementation of the simulated model in chapter 3 into SDR architecture.

Chapter 5: The tests of the implemented model in both real time and simulation are carried out. The results are discussed in this chapter.

Chapter 6: In this chapter the conclusion is given and suggestions are made for future work.

This thesis results in software procedures that works efficiently, was tested thoroughly, and was taken up in the SDR library of the research group. The excellent measured implementation loss of 0.3dB proves the value of the implementation. The fact that both transmitter and receiver algorithms run together at 500 times slower than the real time on a 1600 MHz PC, gives an indication of the execution speed.

The next chapter will now introduce DAB and typical specifications for existing standards.

Chapter 2

LITERATURE SURVEY ON DAB

2.1 Introduction

This chapter provides a theoretical background of the DAB system and practical considerations regarding its implementation. These include the DAB system layout and its operation, transmission signal structure and its characteristics, transmission standard and the transmission modes. The theory on DAB signal modulation and demodulation using COFDM is also covered in this chapter.

2.2 What is DAB?

DAB, Digital Audio Broadcasting, is a digital method of delivering radio services from the studio to the receiver. It is the one of the most significant advances in radio broadcasting technology since the introduction of the Frequency Modulation (FM) stereo radio system. DAB is a completely new radio broadcasting system intended for delivering high-quality digital audio programmes and data services to fixed, mobile and portable receivers, which can use simple antennas.

Broadcast radio has been in widespread use since 1920s, and to this time has remained largely based on the analogue " amplitude modulation"(AM) technologies used at the beginning and the "frequency modulation"(FM) technologies introduced in the mid-20th Century [1]. These analogue radio broadcasts were thought up and designed to serve household receivers (static users) [2] using fixed and directional rooftop antennas. But with the development of new, small and cheaper electronic devices, the majority of radio listening today is carried out with portable and mobile receivers, which use only simple whip antenna. This has resulted in the analogue standard failing to provide many listeners with the audio quality they have come to expect in this age of compact discs, where all audio sources are compared [1] [2]. There is a demand for

something that was not originally part of the broadcast plan: mobile reception. Thus to enable higher fidelity, greater noise immunity and new services the DAB standard had to be developed.

Analogue radio networks are able, of course, to provide good quality radio services for most of the mobile and portable users under favourable reception conditions. When conditions are less favourable, both broadcasts suffer a loss of broadcast quality. Examples of this include [3]: FM reception is badly affected by shadowing and signal reflection from buildings or hills (multipath propagation), and AM systems are affected by seasonal propagation variation that causes fading and occasional loss of signal. These occur because these systems do not provide measures to combat the effects of multipath propagation and interference, which is difficult to do when we are talking about mobile communication environments. The multipath effect is illustrated in Figure 2.1 according to [4].

Based on the point mentioned in the above paragraph, there is little that can be done to rescue traditional analogue broadcast signals (an FM signal or any other analogue signal) in the presence of severe fading and interference. To solve these problems and provide audio broadcasting of compact disc quality [5], the European Eureka project developed a digital audio broadcasting (DAB) system. For example with just a simple non-directional whip antenna, DAB eliminates interference and the problem of multipath, together with wide area coverage with no signal interruption.

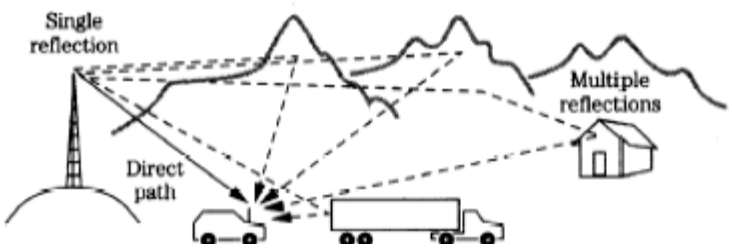


Figure 2.1 Effect of multipath on signal in mobile environment.

The DAB system standard that is discussed in the next sections has been developed within the European Project called Eureka 147 [5]. The standard is commonly referred to as the Eureka

147 digital audio broadcasting standard. It is the European broadcasting standard used for mobile, portable and fixed receivers, and has been standardized by the European Telecommunication Standard Institute (ETSI) [5].

The system standard is designed to deliver high-quality digital sound programmes and data services for both home and portable, but especially for mobile receivers. It includes advanced digital techniques to provide ruggedness, sufficient to combat the effect of multipath propagation, Doppler spread and interference. The Eureka 147 DAB standard is designed to operate in any frequency band in the VHF and UHF range for the terrestrial, satellite, hybrid (satellite and terrestrial), and cable broadcast networks. The standard is acceptable for use as the digital radio standard almost worldwide with the exception of USA and Japan [4] [6]. Japan has developed its own national solution called ISDB-T (Terrestrial Integrated Services Digital Broadcasting) [7]. In the USA the National Association of broadcasting refused to adopt the Eureka-147 standard. The USA adopted a digital radio scheme that use an approach known as In-Band On-Channel (IBOC)[4].

2.3 What DAB offers to the Broadcaster and Listeners

The Eureka 147 DAB system offers both listeners and broadcasters a unique combination of benefits and opportunities in comparison with conventional analogue radio broadcasting [8] [9] [10]. These include:

1) Rugged and reliable delivery of radio services to fixed, portable and mobile receivers, free from interference. This provides a means for a broadcaster to reach listeners with high-quality digital audio services.

2) Efficient use of the limited radio frequency spectrum available. This provides the possibility of increasing the number of radio stations and carrying more radio programmes.

3) An added-value system feature that allows enhancements to existing radio services, for example radiotext, graphics and still-picture.

4) The possibility of constructing Single Frequency Networks (SFNs) [11] [12]. In SFNs, all transmitters covering a particular area broadcast the same information and operate on the

same frequency with contiguous coverage zones. Although the signals emitted by the various transmitters are received with different time delays, the receiver automatically selects the stronger signal without interference from overlapping zones. This eliminates the problem of having to retune a receiver at frequent intervals such as in car, and allows efficient use of spectrum.

5) The provision of a wider choice of programmes for the listener and easy tuning of the receiver.

2.4 The DAB system – How it works

In this section, simplified descriptions of the principles employed in the DAB transmission system to broadcast sound radio services will be discussed. The descriptions are based on the DAB transmission system [5] illustrated in Figure 2.2. The processing stage involved in the generation of the DAB signal together with the signal path through transmitter elements are briefly presented.

The DAB system is made of a number functional blocks (see Figure 2.2) that work together to process the input services and output the DAB transmission signal. In the figure each functional block is labelled according to the function it performs. This enhances a clear understanding of what is going on inside a block and how the system works in general. The system operation is described by a chain of events that follow the signal paths through the DAB transmitter blocks in the left-to-right direction. This chain of events is explained as follows:

a) At the input of the system the analogue signals such as audio and data of the services are encoded, then error protected and time interleaved.

b) The output services in (a) are then combined to form the Main Service Channel (MSC) in the Main Service Multiplexer.

c) The output of the multiplexer is then combined with multiplexer control data and service information in the Fast Information Channel (FIC) to form a transmission frame in the Transmission frame multiplexer (see figure 2.2).

d) Lastly, Orthogonal Frequency Division Multiplexing is applied at the output of the multiplexer to shape the DAB signal made up of a large number of carriers.

The above describes the operation of the transmission system in general, the detail of what is going on in each block is not presented. The reader is referred to [5] for detailed information. But for a clear understanding of how the system works, the generation DAB signal and how the system achieves the advantages presented in section 2.3, the main three system elements [8] are presented in detail. These are:

- Source coding (MUSICAM Audio Coding)
- Multiplexing and Transmission Frame.
- COFDM Modulation.

The first two elements are presented in section 2.5 and 2.6 respectively. Section 2.7 describes COFDM Modulation that is the main part of the DAB system and the main focus of this thesis.

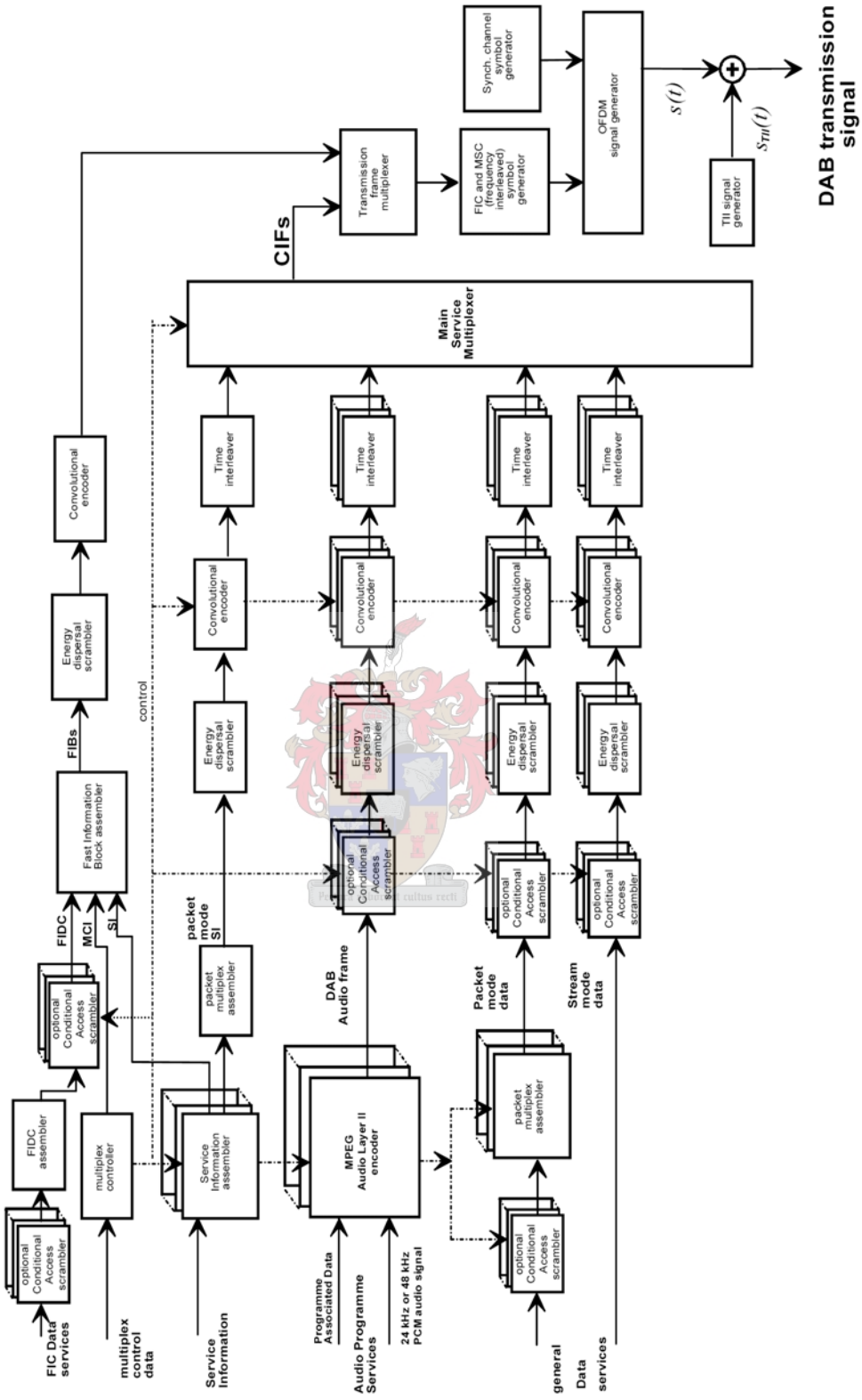


Figure 2.2 DAB transmitter block diagram [5].

2.5 Source Coding (MUSICAM Audio Coding)

According to [13] the available DAB gross bit is about 2.3Mbits/s and, within certain quanta, this can be apportioned to sound-programme data and error protection data as required. However, there is a trade-off between the ruggedness of mobile reception and the programme capacity. The optimum capacity for the terrestrial radio transmission may be approximately equal amounts of error protection and programme data, in which case the capacity is around 1.2Mbits/s. However, the studio standard for digital audio signals prescribed by the AES/EBU interface, uses 16-bit linear PCM with 48kHz sampling rate, so a single full bandwidth (20 Hz to 20kHz stereo audio signal) requires 1.5Mbits/s. A compact disc has a similar requirement. Therefore, it is essential that the bit rate of the sound programme data must first be reduced, and this is the function of a source encoder.

The source encoder used in the DAB system can reduce the required bit-rate by a factor of 6 or more. It employs a digital audio compression technique [14] known as MUSICAM (Masking Pattern, Universal Sub-band, Integrated Coding And Multiplexing). The technique processes the input linear Pulse Code Modulation (PCM) audio signal (see Figure 2.2) sampled at 48kHz or 24kHz, and produces the compressed audio bit stream [15] of different bit rates ranging from 8kbit/s to 384kbit/s.

MUSICAM employs [16] the method of psycho acoustical coding specified for MPEG-2 Audio Layer II encoding. This exploits the knowledge of the properties of the human auditory system. The technique codes only audio signal components that the ear will hear, and discards any audio information that according to the pyschoacoustical model, the ear will not perceive, an example of these includes very quiet sounds that are masked by the other and louder sounds. So, using this method the bandwidth is allocated only to the essential information that derives a high quality signal. This allows DAB system to use a spectrum more efficiently and to deliver high quality audio signal to the listener.

2.6 Multiplexing and Transmission Frame

In section 2.4, it was presented that data for individual services such as audio, or data are to be initially encoded at individual level, error protected and time interleaved. The output services are then combined into a single data stream ready for transmission. The process of combining data stream is known as multiplexing and the resulting data stream is called the multiplex.

In order to facilitate receiver synchronization, the DAB signal [5] is designed according to the frame structure with a fixed sequence of symbols illustrated in Figure 2.3.

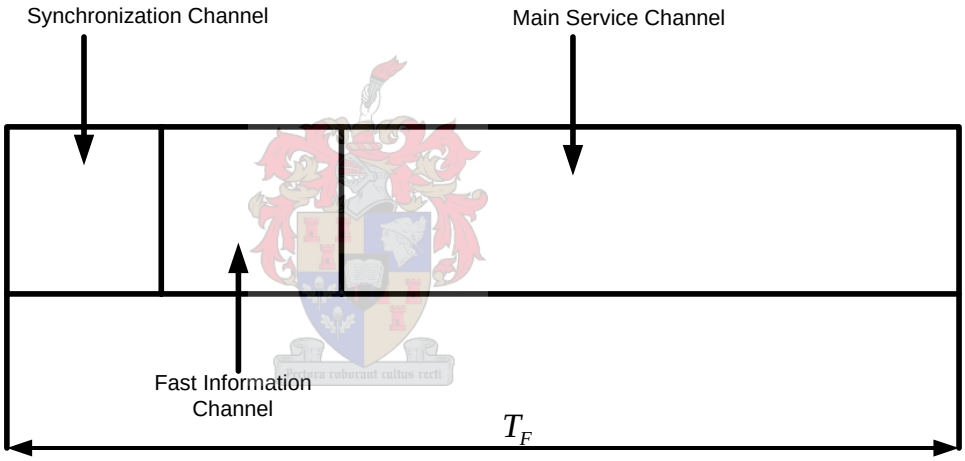


Figure 2.3 DAB transmission frame structure

Each DAB transmission frame has duration of T_F, and comprises of the three distinct channels explained below:

1) The Main Service Channel (MSC) is the logical channel where the information of the programmes is carried (audio and data service components). It is a time-interleaved data channel divided into a number of sub-channels, which are individually convolutionally coded with equal or unequal error protection. Each sub-channel may carry one or more

service components. The organization of the sub-channel and service components is called the multiplex configuration.

2) Fast Information Channel (FIC) is used for rapid access information by a receiver. In particular it is used to send the multiplex configuration information and optional service information and data service. The multiplex configuration information enables the receiver to decode the signal correctly. The FIC is a non-time-interleaved data channel that is highly protected to ensure its ruggedness. The FIC is made up of a number of Fast Information Blocks (FIB's). Depending on the transmission mode used, different numbers of FIB's are multiplexed in one transmission frame to form the FIC. The FIC forms three consecutive blocks of the DAB transmission frame.

3) A synchronization channel comprises two symbols. One is the null symbol, which is the duration of no RF signal transmitted, and the other symbol is a phase reference symbol, which has a predetermined modulation. The channel is used internally within the transmission system for basic demodulator functions, such as transmission frame synchronization, automatic frequency control, and channel state estimation and transmitter identification. This allows effective receiver synchronization and decoding of the received DAB signals.

In Figure 2.3 it is important to note that each transmission frame begins with a null symbol for a coarse synchronization when no RF signal is transmitted, followed by a phase reference symbol. The next three symbols are reserved for the FIC and the remaining symbols provide MSC. The total frame duration, T_F is 96ms, 48ms or 24ms depending on the transmission mode (see Section 2.9). The multiplex data is distributed amongst the entire carriers, occupying 1.54MHz spectrum.

2.7 COFDM Modulation

Digital audio broadcasting has the potential to give every radio the sound quality of a compact disc. To accomplish this, it requires a rugged method of transmission. The Coded Orthogonal Frequency Division Multiplexing (COFDM) modulation system was developed to meet this need. This is the heart of the Digital audio broadcasting. The modulation scheme uses many

carriers, up to 1536, spaced at 1kHz, where each carrier is independently modulated using Differential Quadrature Phase Shift Keying (D-QPSK).

The COFDM combines a multi-carrier modulation technique OFDM (Orthogonal Frequency Division Multiplexing) together with an error-correcting code (Convolutional channel coding). The detail of each is described in the next subsections with OFDM described in section 2.7.1 and the used error correction code described in sections 2.7.4.

2.7.1 OFDM

OFDM is a multi-channel modulation scheme employing Frequency Division Multiplexing (FDM) of orthogonal carriers, which makes the 'Orthogonal' part of COFDM. It spreads the data to be transmitted over a large number of closely spaced carriers. Only a small amount of the data is carried on each carrier. So the data rate to be conveyed by each carrier is correspondingly reduced.

In OFDM signal, the carriers have a common frequency spacing that is precisely chosen. This is an inverse of the duration called the active symbol period (T), over which the receiver will examine the signal and perform demodulation. The choice of the carrier spacing (1/T) ensures that all carriers are mathematically orthogonal to each other. Thus the spectrum of each carrier is null at the centre frequency of the other carriers in the system [17], this is illustrated in Figure 2.4.

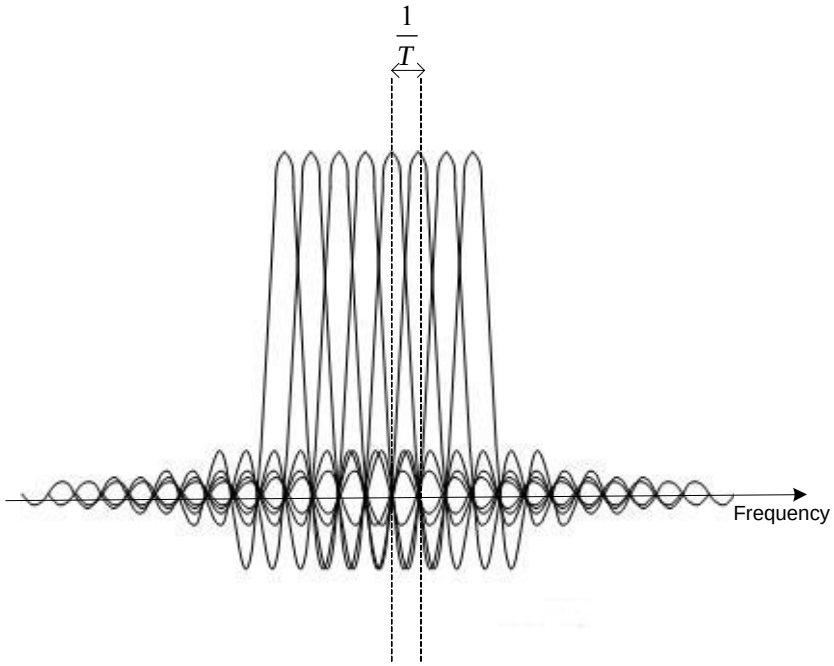


Figure 2.4 OFDM spectrum

To understand the concept of orthogonality presented in the above paragraphs, let us consider a set of signals Ψ, where Ψ_p is the pth element in the set. The signals are mathematically orthogonal if:

$$\int_a^b \Psi_p(t)\Psi_q^*(t)dt = K \quad for\ p = q$$
$$= 0 \quad for\ p \neq q \tag{2.1}$$

where the * indicates the complex conjugate.

The orthogonality enables each carrier in the OFDM system to be extracted from the set with no interference from the other carriers, since each one of the carriers is positioned in one of the zero energy frequency points of all of the other carriers (see Figure 2.4). This means carriers can be generated and recovered without carrier specific filtering.

Fortunately the apparently very complex processes of modulating (and demodulating) large numbers of carriers simultaneously are equivalent to Discrete Fourier Transform (DFT) operations, for which efficient Fast Fourier Transform (FFT) algorithms exist. The Fast Fourier Transform (FFT) can be implemented very efficiency in electronic hardware or software. This makes OFDM implementation feasible.

2.7.2 The use of FFT in COFDM

In section 2.7.1 the concept of orthogonality of an OFDM has been discussed. The application of this makes it possible to split bits into two orthogonal components, called the In-phase (I) and Quardature components (Q). The bits can be handled like a complex number, where the real part would be I-component and imaginary part the Q-component. The whole signal could be transmitted in a parallel way with a two-shifted version of the same carrier (sine and cosine), using complex modulation.

The COFDM technique has taken so long to come into prominence because of the practical reasons [18] such as the need of the large number of sub-channels and the array of sinusoidal generators and coherent demodulation required in a parallel system (see Figure 2.5). It has been very difficult to generate a signal, and even harder to receive and demodulate the signal. The hardware solution, which makes use of multiple modulators and demodulators in parallel, was somewhat costly, complex and impractical for use in a domestic system. Figure 2.5 shows an example of array of sinusoidal generators used in a multicarrier system.

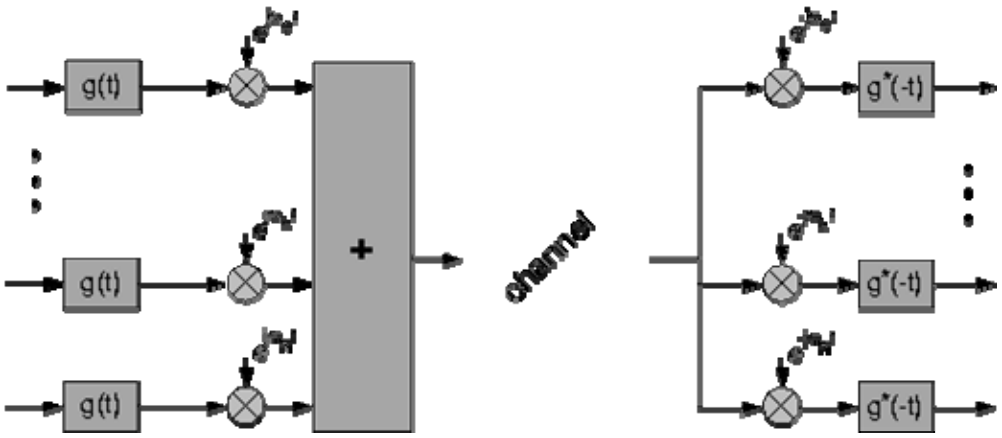


Figure 2.5 Basic structure of a multicarrier system.

In 1971 Weinstein and Elbert [19] suggested the application of the Discrete Fourier transform (DFT) to parallel data transmission systems as the part of the modulation and demodulation process. This eliminated the bank of sub-carrier oscillator and coherent demodulators required. Thus the signal is defined in the frequency domain and is generated using inverse DFT. At the receiver the reverse process is used. Both DFT and IDFT are implemented using Fast Fourier Transform (FFT) algorithms.

The Fast Fourier Transform is merely a rapid mathematical method for calculating the DFT. It is the availability of this technique and technology that allow it to be implemented in integrated circuits at a reasonable price, that has permitted COFDM to be developed as far as it has. Using very large scale integration (VLSI) and digital signal processing (DSP) technologies have reduced the implementation cost of OFDM systems drastically. The inverse FFT provides a series of digital samples, which are the time domain representation of the signal. Figure 2.6 shows a block diagram of OFDM system according to [20].

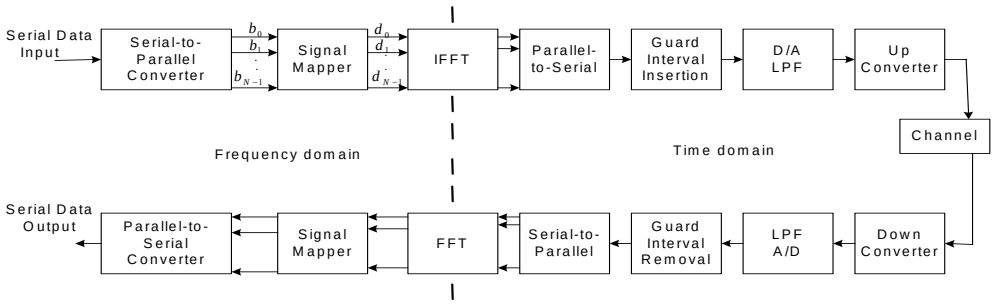


Figure 2.6 FFT-based OFDM system

Figure 2.6 illustrates the process of a typical FFT-based OFDM system. The incoming high-speed serial data is first converted from serial to parallel (N low speed data stream). Each of these low data streams is grouped into x bits to form a complex number (mapping output). The number x determines the signal constellation of the corresponding sub-carrier, such as PSK, QPSK, 16 QAM or 32 QAM. The complex numbers are modulated in baseband fashion by inverse FFT and concatenated to serial data for D/A conversion. A guard interval is inserted between symbols to avoid Inter-Symbol Interference (ISI). The discrete symbols are concatenated, converted to analogue and low pass filtered for RF up conversion. The receiver performs the inverse of the transmitter.

After the qualitative description of the OFDM system it is valuable to discuss the mathematical definition of the system. [19] [21] shows how this can be done mathematically (see below).

Consider a data sequence d_o, $d_1 \ldots d_{N-1}$, where each d_n is a complex symbol. The data sequence could be the output of a digital modulator, such as QAM, PSK QPSK etc.

The complex symbol d_n can be expressed as:

$$d_n = a_n + jb_n \tag{2.2}$$

where $a_n = \cos\phi_n, b_n = \sin\phi_n$ *and* ϕ *is the phase.*

The waveform of an individual sub-carrier at frequency nf_0 can be defined as:

$$x_n(t) = \sqrt{a_n^2 + b_n^2}\cos(2\pi(f_c + nf_o)t + \phi_n)$$

$$= a_n \cos(2\pi(f_c + nf_0))t - b_n \sin(2\pi(f_c + nf_0))t \tag{2.3}$$

where

$$f_0 = 1/T \qquad and \qquad \phi_n = \tan^{-1}\frac{b_n}{a_n} \tag{2.4}$$

f_c is the central frequency of the signal.

When this is summed over all N sub-carriers, the generated OFDM signal is:

$$x(t) = \sum_{n=0}^{N-1}\left[a_n \cos\{2\pi(f_c + nf_0)t\} - b_n \sin\{2\pi(f_c + nf_0)t\}\right] \tag{2.5}$$

In (2.5), it seems as if the N of digital modulators and the N of sub-carriers generator are required. This is too much to implement.

But (2.5) can be written as:

$$y(t) = \mathrm{Re}\left\{\sum_{n=0}^{N-1} d_n \exp(j2\pi(f_c + nf_0)t\right\}$$

$$= \mathrm{Re}\left\{\left[\underbrace{\sum_{n=0}^{N-1} d_n \exp(\frac{j2\pi kn}{N})}_{IDFT}\right]\exp(j2\pi f_c t)\right\} \tag{2.6}$$

where

$$t = \frac{k}{Nf_0} \tag{2.7}$$

The terms enclosed in the square bracket (2.6) define an IDFT and represent the base-band version of the OFDM signal. In the receiver the inverse of the transmitter process is applied.

The transformation of (2.6) requires N^2 complex products. In order to work with real time systems, it would be useful to handle the complex signals as quickly as possible. The method to work faster with the DFT [22] [23] [24] is the FFT/IFFT algorithm, which is the main part of the DAB transmission system. The FFT reduces the number of computations to the order of $N/2*\log_2 (N)$. To enable the signal to be generated using inverse FFT, it is preferable that the number of carriers considered in the calculation is an integer power of two. In practice, it is not always desirable to have the number of the real carriers restricted in this way. However, it is convenient to make up the actual number of those to a power of two by setting the amplitude of those not wanted to zero.

In this subsection it has been shown that an OFDM scheme uses minimum frequency spacing between sub-carriers, its use in DAB makes the system use the precious spectrum more efficient. Also the use of N parallel channels in OFDM has the effect of increasing symbol duration and so reduce the effect of Inter Symbol Interference (ISI). To further mitigate the effect of ISI, DAB system uses guard intervals between consecutive OFDM symbols. The use of the guard interval and its implementation is described in following subsection.

2.7.3 Guard interval and its implementation

In order to overcome the problem of multipath propagation especially in mobile receivers, DAB adds a guard interval between OFDM symbols. The guard interval is formed by a cyclic continuation of the signal [5], so that the information in the guard interval is actually present in the OFDM symbol. The added interval extends the total length of the transmitted symbol by approximately one quarter of the symbol length. The guard interval is added by taking a copy of the last portion of the OFDM symbol and putting it at the start of the symbol. This effectively extends the symbol, while maintaining orthogonality of the waveform, which essentially prevents one sub-carrier from interfering with another (called inter-carrier interference, or ICI). Figure 2.7 illustrates the use of guard interval and its implementation.

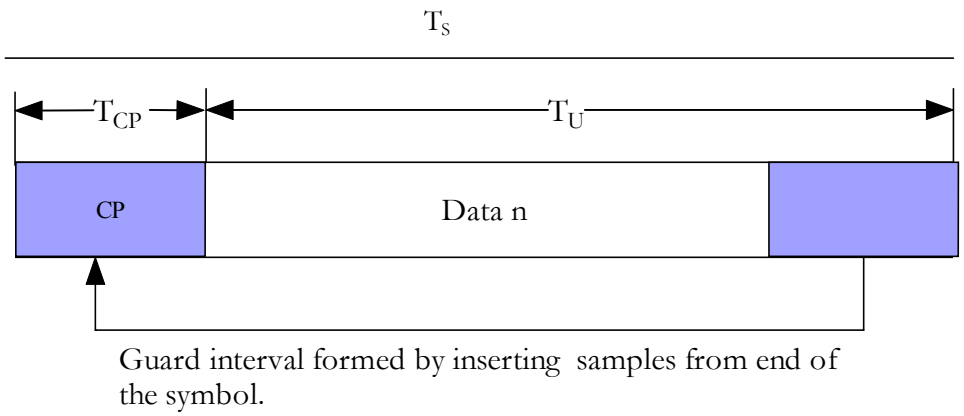


Figure 2.7: Guard interval and Cyclic Prefix

Where T_U is the OFDM symbol time without guard interval, T_{CP} is the duration of the copied information in the guard interval using cyclic prefix and T_S is the total OFDM symbol duration.

Using cyclic extension and given the fact that phase difference carries the information, the samples required for performing the FFT (decoding the symbol) can be taken anywhere over the length of the symbol. This provides multipath immunity as well as symbol time synchronization tolerance.

The DAB system sizes the cyclic prefix appropriately to serve as a guard time to eliminate ISI. This is accomplished because the duration of the cyclic prefix used in the system is greater than the amount of time dispersion from the channel [25]. The values of the guard period for each transmission mode are given in Table 2.1.

2.7.4 Error correcting code (Convolutional channel coding)

The use of OFDM in the DAB system provides a very good basis for rugged receptions under multipath conditions but further measures are necessary to realise the full system benefits. On its own, OFDM with a guard interval can be used to minimise the effect of ISI. However, ISI is a time domain effect [26], and multipath propagation has effect in the frequency domain, which may result in the partial or total cancellation of some frequencies at the receiver. The DAB system attempts to eliminate this effect with the use of error correction code (convolutional channel coding). This accounts for the 'Coded' part of the name COFDM. The punctured

convolutional coding is used [5]. This adds redundancy to the data in order to help the receiver detect and better eliminate transmission errors.

The error correction process works best if the errors in the incoming data are random. To ensure this the transmitted data in the Eureka 147 DAB system is interleaved over all the carriers and over a range of time. These are used together to combat the effect of frequency selective fading.

The interleaving is a process involving the re-ordering of the bits-stream in the transmitter before using it to modulate the carriers. The idea is to distribute the signal over all the carriers and so to spread the information symbols. As the result, if the specific carrier fades away, it will cause some error bits in several blocks' symbols and not many error bits in only one symbol. So the channel coding will be able to correct the wrong data by using the correct information that is present in the rest of the symbols, thanks to the rest of the frequency carriers that were not fading.

2.8 DAB transmission signal

After discussions on how the DAB system works as presented in the above sections, it is convenient to define the DAB transmission signal according to [5]. The DAB main transmission signal is made up of a numbers of transmission frames as discussed in section 2.6. Each transmission frame is divided into a sequence of OFDM symbols, each made up of a fixed number of carriers. The number of OFDM symbols in a transmission frame depends on the transmission mode, as will be defined in the section 2.9. The carriers in each OFDM symbol are equally spaced with the carriers' frequency spacing equal to the inverse of the useful symbol duration (T_U).

According to the system standard, the first two OFDM symbols of any transmission frame are made up of a synchronization channel regardless of the transmission mode (see Figure 2.3). The standard defines the first OFDM symbol for each transmission frame to be a Null symbol of duration T_{NULL} and the remaining part of the frame to be made of OFDM symbols of the duration T_S. The symbol duration T_S comprises of the useful symbol duration T_U and a guard

interval with a duration Δ (see Figure 2.7). The DAB signal occupies a bandwidth of 1.536MHz and uses a large number of discrete carriers, each independently modulated, using π/4 D-QPSK.

The defined main DAB transmission signal s (t) [5] is given in the formula below:

$$s(t) = \mathrm{Re}\left\{ e^{2j\pi f_c t} \sum_{m=-\infty}^{\infty} \sum_{l=0}^{L} \sum_{k=-K/2}^{K/2} z_{m,l,k} \times g_{k,l}(t - mT_F - T_{NULL} - (l-1)T_S \right\} \quad (2.8)$$

With,

$$g_{k,l}(t) = \begin{cases} 0 & for \quad l = 0 \\ e^{2j\pi k(t-\Delta)/T_U} . \mathrm{Rect}(t/T_S) & for\ l = 1,2,...,L \end{cases} \quad (2.9)$$

and $T_S = T_U + \Delta$.

where,

L is the number of OFDM symbols per transmission frame (the Null symbol being excluded);

K is the number of transmitted carriers;

T_F is the transmission frame duration;

T_{NULL} is the Null symbol duration;

T_S is the duration of OFDM symbol of indices l=1,2,3,…, L;

T_U is the inverse of the carrier spacing;

Δ is the duration of the time interval called guard interval;

$z_{m,l,k}$ is the complex D_QPSK symbol associated with carrier k of OFDM symbol l during transmission frame m. For k=0, $z_{m,l,k}$=0, so that the central carrier is not transmitted;

f_c is the central frequency of the signal.

These parameters are specified in Table 2.1 for each transmission mode [5], which is in the next section.

If we consider equation 2.8 for the period from t=0 to t=T_S, we obtain:

$$s(t) = \left\{ e^{j2\pi f_c t} . \mathrm{Re}\, ct(t/T_S) \sum_{k=0}^{K} z_{o,1,k} e^{j2\pi k'(t-\Delta)/T_U} \right\} \tag{2.10}$$

with k'=k-K/2.

There is a clear resemblance between (2.10) and the Inverse Discrete Fourier Transform (IDFT)(2.6). Thus the DAB transmitted signal in the time domain is generated using an inverse FFT algorithm that is the heart of the DAB transmission system. Its convenient implementation is by generating N samples X(n) corresponding to the useful part T_U long, of each OFDM symbol and adding the guard interval by taking copies of the last NΔ/ T_U of these samples and appending them in front of the symbol. A subsequent up-conversion then gives the real signal s(t) centred on the frequency f_c.

2.9 DAB transmission modes

In order to ensure that the DAB system is applicable in different transmission network configurations and over wide range of frequencies, four different transmission modes have been defined, each having its particular set of parameters. These take into account the spectrum availability in the intended frequency range from 30MHz to 3GHz and the practical implementation factors (e.g. the size of the antenna), that a single mode couldn't do.

The system modes defined have the same system capacity of 1.536MHz signal bandwidth [25], but the symbol period (and guard interval) and carrier frequency spacing are varied to suit the situation. In addition, all modes retain the reciprocal relationship between the symbol duration and the carrier frequency separation in order to maintain orthogonality and spectral efficiency. The features of all four modes [5] are summarised below in Table 2.1. All the durations in Table 2.1 are time-related in whole multiples of the elementary period T=1/2 048 000 seconds.

Transmission mode	I	II	III	IV
Number of carriers (K)	1536	384	192	768
Number of OFDM symbols/frame (L)	76	76	153	76
Transmission frame duration (T_F)	196 608 T 96 ms	49 152 T 24 ms	49 152 T 24 ms	98 304 T 48 ms
Null symbol duration (T_{NULL})	2656 T ~1,297 ms	664 T ~324 μs	345 T ~168 μs	1328 T ~648 μs
Total symbol duration (T_S)	2552 T ~1,246 ms	638 T ~312 μs	319 T ~156 μs	1276 T ~623 μs
Useful symbol duration (T_U)	2048 T 1 ms	512 T 250 μs	256 T 125 μs	1024 T 500 μs
Guard interval duration (Δ)	504 T ~246 μs	126 T ~62 μs	63 T ~31 μs	252 T ~123μs
Carrier frequency separation	1kHz	4kHz	8kHz	2kHz

Table 2.1 Characteristics of the four DAB transmission modes

Mode I is intended for terrestrial transmission, particularly using Single Frequency Networks (SFNs) operating at frequencies below 300MHz.

Mode II is intended principally for terrestrial transmission using an individual transmitter (local and regional radio services) at frequencies below 1.5 GHz. Also SFN implementation is possible.

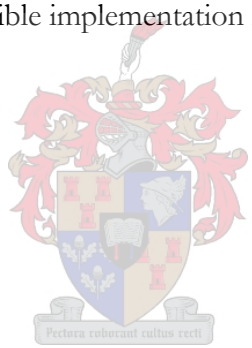
Mode III is intended for cable delivering and satellite-and-complementary terrestrial transmission at frequencies below 3GHz.

Mode IV is used in L-band and allows greater transmitter spacing in SFNs.

2.10 Conclusion

The chapter discusses the background theory that is used in the practical implementation of the DAB in Software Defined Radio. The DAB signal modulation and demodulation using the OFDM scheme together with the required transmission signal structure was discussed. The equation used in the generation of the DAB transmission signal is given in equation 2.8. The standard parameters defining the transmission signal for each transmission mode are given in Table 2.1.

The physical modulation part of the DAB transmitter for the transmission Mode II as well as its' receiver will be implemented. The DAB transmission signal described in section 2.8 and the standard parameter for Mode II discussed in section 2.9 will be used in the implementation. The next chapter describes the simulation details of the implemented DAB system model. The simulation result shows a negligible implementation loss.



Chapter 3

SIMULATION

3.1 Introduction

This chapter describes the simulated model of the physical modulation part of the Digital Audio Broadcasting (DAB) system before its real time implementation in SDR. In the simulation the physical modulation part of the DAB transmitter and one of the receiver models were simulated. The simulation follows the standard parameter specified in the second chapter.

The DAB transmission mode used in the simulation is mode II. This mode has been chosen for the simulation because of its suitability in the local area terrestrial broadcasting to be a model that presents other transmission mode implementation. All the work developed in the simulation model follows this mode standard parameter. The specific numeric values of the parameters that develop the DAB transmission signal are according to section 2.9 in the second chapter.

The complete DAB transmission system comprises of many blocks (see Figure 2.2 and Figure 3.1). The work of this thesis starts in the last part of the transmission system, from the end of the transmission frame multiplexer (see Figure 2.2). The simulation starts from the block partitioner (see Figure 3.1) followed by the modulation, thus the channel coding and time interleaving are not included in the simulation.

The simulation has been developed in base-band transmission. The RF section for both transmitter and receiver was not studied in this thesis. So on the transmitter side neither digital-to-analogue conversion nor quadrature modulation (RF section) were simulated. Similarly on the receiver side neither analogue-to-digital conversion nor quadrature demodulation (RF section) were simulated.

The simulation was done to provide a proper direction before real time implementation and development of working real time software.

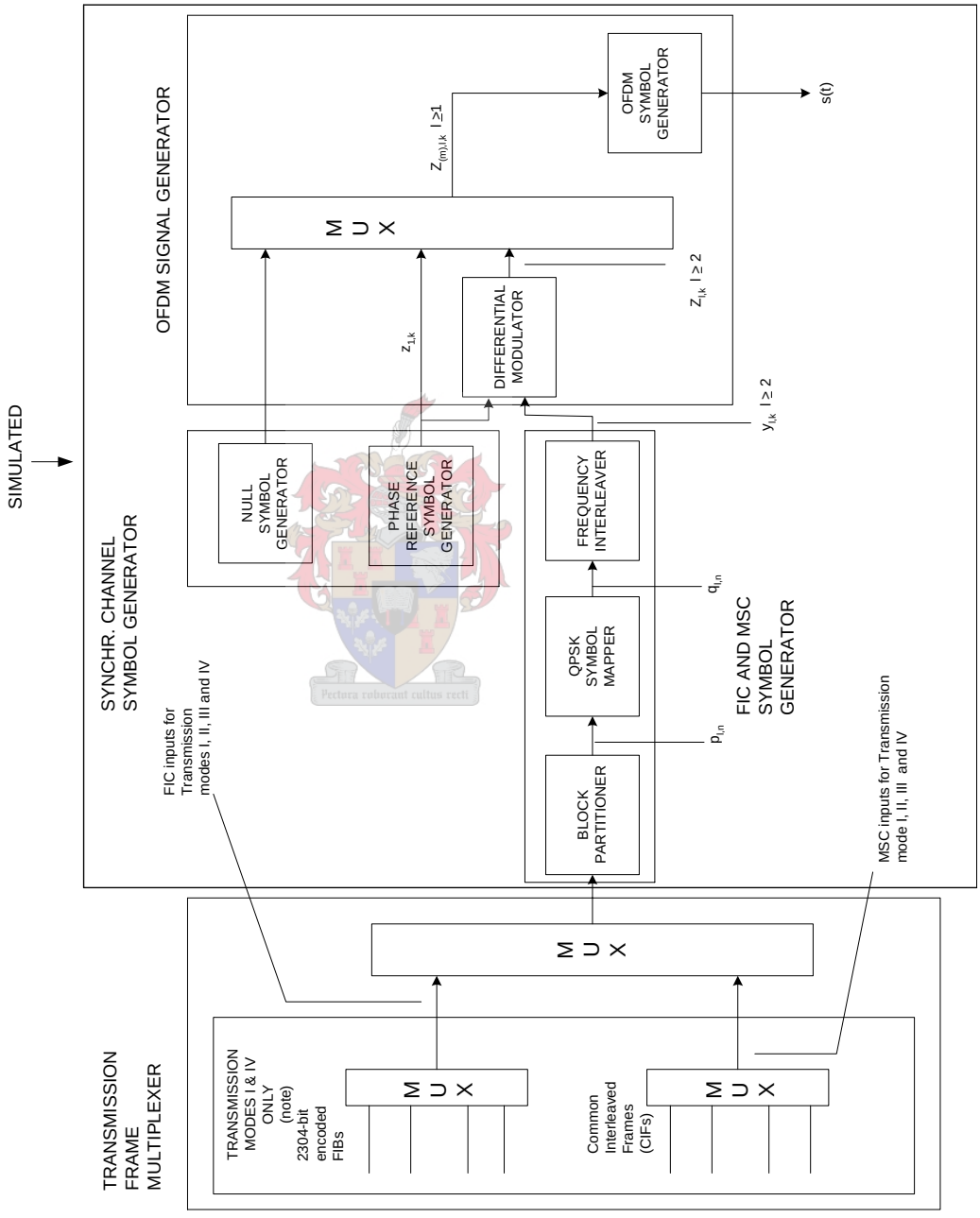


Figure 3.1 DAB Transmission scheme [5]

3.2 Simulation system model

For performing the simulations, the chain shown in Figure 3.2 was developed under MATLAB 6.5 environment. Each block in the figure has its own functionality, which will be discussed in details in the next sections. The MATLAB code implementing each block is shown in Appendix A.

The following is the general overview of the operation of the system:

1. Generate a binary message of random bit sequence with a length equal to one frame size.
2. Partition the generated random bits into data blocks, perform the QPSK symbol mapping to each data block and apply frequency interleaving to the QPSK symbols on each data block. Note a data block constitutes an OFDM symbol.
3. Generate the phase reference symbol and perform the differential modulation on each data block.
4. Add a phase reference symbol at the beginning of the frame, apply zero padding on each data block and perform an inverse FFT operation to each data block.
5. Add cyclic extension to each OFDM symbol.
6. Generate a null symbol and add it at the beginning of the frame from (5) to make a complete frame ready for transmission.
7. Pass it through the channel with additive white Gausian noise.
8. Perform the receiver synchronization.
9. Remove the cyclic prefix from each OFDM symbol of the synchronized received signal, and perform FFT on each OFDM symbol to recover the data signal.
10. Perform zero padding removal from each OFDM symbol.
11. Perform the differential demodulation.
12. Perform the frequency de-interleaving followed by QPSK symbol de-mapping.
13. Calculate the bit error rate of the system.

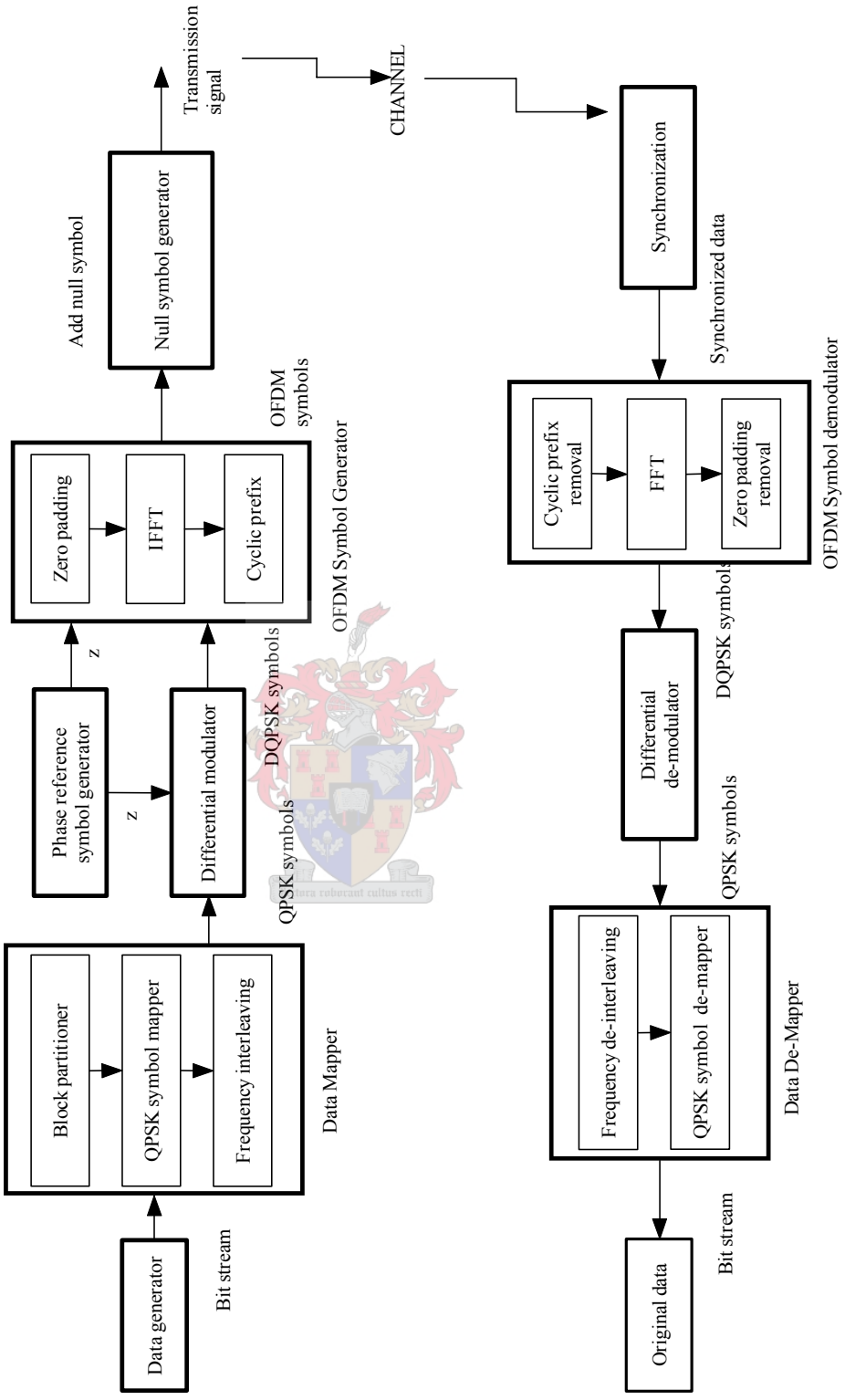


Figure 3.2 Block diagram of the system simulated

3.3 Data generator

The data generator block is the first block in the transmission side. It generates a binary data message that will be transmitted over the system. The random bit sequence generated constitutes a transmission frame data for FIC and MSC in a similar way to that described in section 2.6. This provides inputs to the data mapper block.

The data size for FIC and MSC is known, so the total random bits generated for one transmission frame is given by the following expression:

total_bit = fic_bit + msc_bit (3.1)

Where fic_bit denotes the total random bits for FIC equal to 2304 sample bits and msc_bit denotes the total random bits for MSC equal to 55296 sample bits. This has been calculated from the parameters given in Table 2.1. The number of carriers for mode II is 384 and there is 2-bit per carrier with QPSK modulation. This makes a total bits per OFDM symbol equal to 768. In each transmission frame there are 3-OFDM symbols for FIC and 72-OFDM symbols for MSC. This provides the numeric value for each channel as given above.

The MATLAB function "randint" has been used to generate a stream of random data bits. The function generates random integers that are either 0 or 1 with equal probability. The MATLAB code used to generate a sequence of random bits is shown in next expression:

```
inf_data=randint(1,total_bit)
```
(3.2)

where inf_data presents an array of random bits generated.

Following the transmission mode II, the data generator should generate 57600 bits for each transmission frame.

3.4 Data mapper

The DAB transmission signal described in section 2.8 is made of numbers of OFDM symbols, which are generated using an inverse FFT that works with complex numbers. The data mapper block is responsible for dividing the generated bit array into data blocks, mapping bits in each data block into QPSK symbol constellation and performing frequency interleaving on the QPSK symbols for every QPSK symbol block. Each data block has bit that constitute information for a particular OFDM symbol.

The block performs its task in a sequential order, starting with partitioning the bits array into data blocks, followed by mapping bit in each data block into QPSK symbols and ending with interleaving the symbols in the data block after QPSK mapping (see Figure 3.3). The output of the block is the sequence of symbols (i.e. complex numbers) that describes the input bits being converted into phase.

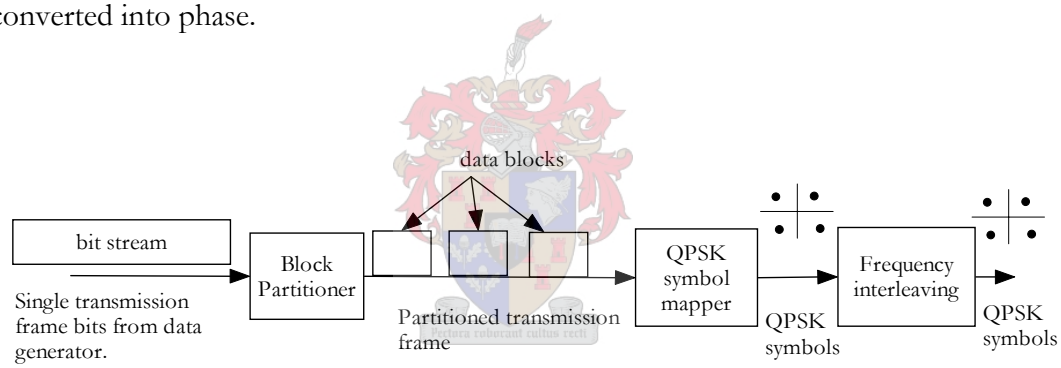


Figure 3.3 Data mapping process

The data mapping consists of a five-processing chain:

- Get array of bit
- Partition array of bits into data blocks
- Perform QPSK symbol mapping
- Perform frequency interleaving
- Store QPSK symbol in the array

The three main processes of the data mapper are block partitioning, QPSK symbol mapping and frequency interleaving. These three main processes form three sub-blocks of the data

mapper. The details of these sub-blocks will be discussed in sections 3.4.1, 3.4.2 and 3.4.2 respectively. Figure 3.4 shows a flowchart that implements the simulated data mapper.

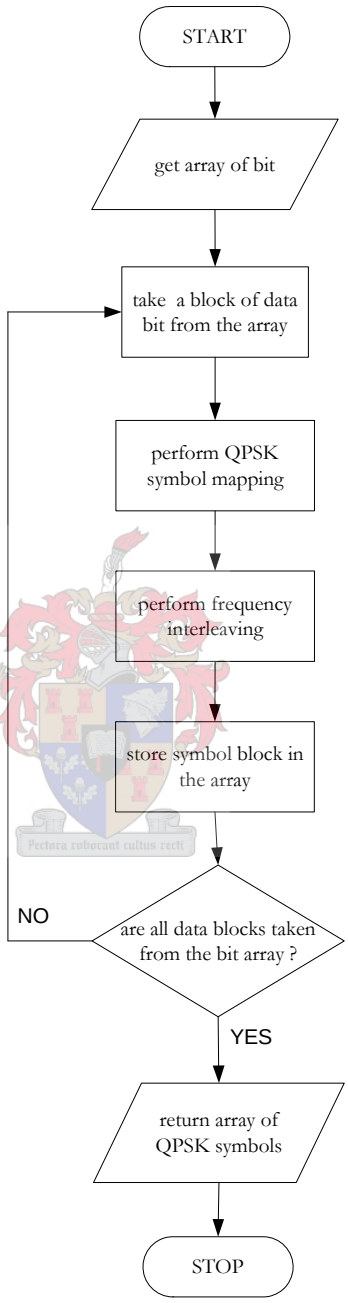


Figure 3.4 Data mapper flow chart

3.4.1 Block partitioner

The generated bit array at a given time constitutes the data bits for a single transmission frame. From equation 2.8 a transmission frame is made of a number of the OFDM symbols and each OFDM symbol is made of a number of carriers. This means that the generated bits in the array should be associated with the OFDM symbols and information (phase) must be assigned to each carrier. In order to achieve this, the generated bit array has to be divided into groups of bit sequences, where bit in each group will constitute an OFDM symbol. To accomplish this task a block partitioner is required. This block divides the bit arrays into data blocks that contain a certain sequence of bit from the generated bit array as described in the next paragraphs.

An OFDM symbol is made of 384 carriers (mode II) and each carrier is assigned a QPSK symbol made of two bits. Each data block will contain 768 bits. The block partitioner divides the input's bit array into data blocks each with 768 sample bits and passes each data block to the QPSK symbol mapper block at a different time interval (see Figure 3.3 and 3.4). So the array of 57600 bits generated is logically divided into 75 data blocks that form 75 consecutive OFDM symbols of index l=2,3…76 in the transmission frame (see section 2.8 and 2.9).

Figure 3.5 illustrates how the array of generated bits is logically divided into data blocks. In the figure each data block is associated with an OFDM symbol. The index l of an OFDM symbol starts at l = 2, because the first OFDM symbol (index l=1) in the transmission frame is reserved for the Phase Reference symbol.

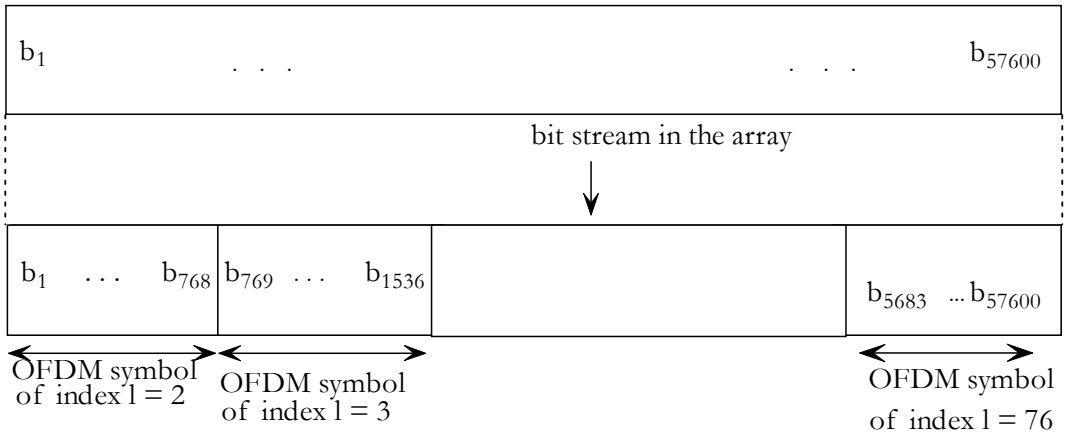


Figure 3.5 Principle of block partitioning

3.4.2 QPSK symbol mapper

The QPSK symbol mapper block is responsible for mapping serial bit streams in each data block into QPSK symbol constellation. So zeros and ones are converted into phase (see Figure 3.7).

A series of 768 bits in each data block is mapped in parallel into a digital constellation according to the QPSK modulation scheme, where two bits in the data block are grouped together and mapped to one of the four symbols in the constellation (see Figure 3.7). This results in generating two data streams, called In-phase and Quadrature (I and Q). The symbol mapping is according to the DAB mapping standard [5] defined next:

$$q_n = \frac{1}{\sqrt{2}}[(1 - 2.b_n) + j(1 - 2.b_{n+K})] \tag{3.3}$$

for n=1,2,…,K

where q_n is the complex QPSK symbols generated with two bits b_n and b_{n+K}, (the value of b can be either 1 or 0) and K is the total number of carriers used.

The data bits in each data block are mapped into 384 (K) complex QPSK symbols. The first QPSK symbol (q_1) is formed with bit-pair bit b_1 and bit b_{385} from data block, the second QPSK symbol (q_2) is formed with bit-pair bit b_2 and bit b_{386} from data block, and so on. The first bit in each bit-pair (b_n) is used to generate I-component and the second bit (b_{n+K}) is used to generate the Q-component of the generated symbol stream. Each bit-pair is referred to as a symbol (S) and each symbol forms one complex QPSK symbol (q_n) defined in (3.3). Figure 3.6 illustrates how bits in each data block are combined to form complex QPSK symbol q_n.

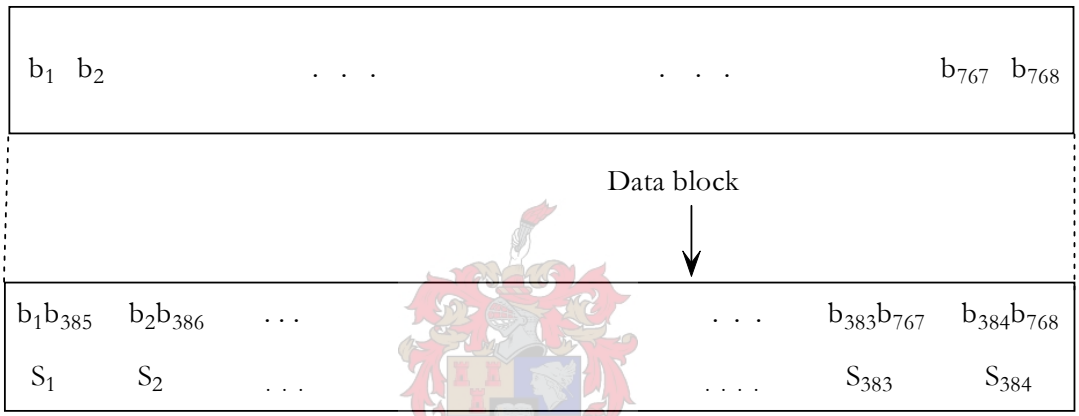


Figure 3.6 Bit-pair forming a complex QPSK symbol array.

According to the mapping, symbol (01) has a positive real part and a negative imaginary part, symbol (10) has a negative real part and a positive imaginary part, symbol (00) has both positive real part and an imaginary part and symbol (11) has both negative real part and an imaginary part. These mapping features are illustrated in Figure 3.7 and bit-pair mapping is shown below:

Bits	Phase
00	45^0
01	-45^0
10	135^0
11	-135^0

The differential part of the QPSK will be discussed in section 3.6.

Figure 3.7 shows the simulated QPSK constellation mapping

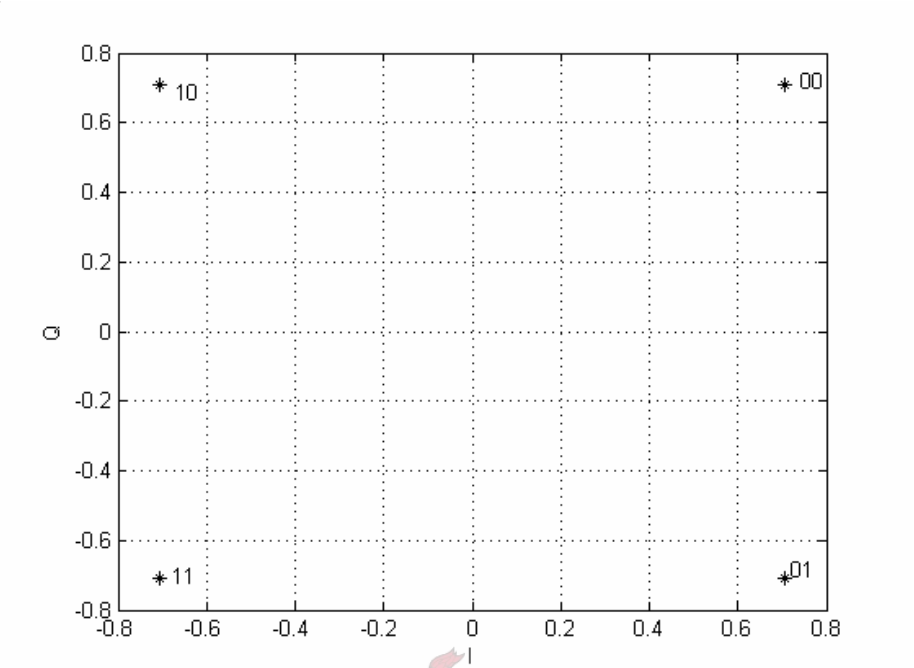


Figure 3.7 QPSK constellation mapping

3.4.3 Frequency interleaving

This section describes the implementation of the interleaving introduced in section 2.7. The frequency interleaving for mode II is described in detail (how it works and how it has been implemented).

The frequency interleaving defines the correspondence relation between the QPSK symbol index n of the QPSK symbols obtained in section 3.4.2 and the carrier index k ($-K/2 \leq k < 0$ and $0 < k \leq K/2$) defined in section 2.8. Each of the 384-QPSK symbols to be transmitted is given an index n, 1 to 384 and each of 384 carriers is given an index k, -192 to 192, omitting 0, which corresponds to the un-modulated center carrier (because phase of DC can not be modulated). The relation between the QPSK symbol index n and the carrier index k, is established by re-ordering the QPSK symbols in the array according to the relation described next. This help to ensure that successive source samples are not affected by selective fade.

To explain how the frequency interleaving works in DAB system [5], let's consider the example of the transmission mode II used in the simulation. The relationship between the input and the output of the frequency interleaver is described by the following expression:

$$y_k = q_n \quad with \quad k = F(n) \tag{3.4}$$

where y denotes the outputs of the interleaver and q_n denotes an input QPSK symbol array to the interleaver (see Figures 3.4 and 3.6). Note the input QPSK symbol array to the interleaver is made of 384 QPSK symbols.

The carrier index k in equation 3.4 is obtained from the index n using mathematical expression $F(n)$ defined in the following paragraphs.

Let $\prod(i)$ be a permutation in the set of integers i = 0,1,2…511, obtained from the next relation:

$$\prod(i) = \left[13\prod(i-1) + 127\right](\mathrm{mod}\,512) \; and \; \prod(0) = 0; \quad for \; i = 1,2,...,511. \tag{3.5}$$

Let D be the set D={d_0, d_1, d_2,…, d_{383}}, containing 384 elements in the same order as $\prod(i)$, but excluding the elements of $\prod$ which are not in the range [64 448] and excluding 256.

So $d_n = \prod(i)$ in the range [64 448] excluding 256.

The correspondence between the index n Є {0,1,2,…, 383} of the QPSK symbol q_n and the frequency index k Є {-192, -191, -190 ,…,192}\ {0} is given by :

$$k = F(n) = d_n - 256 \tag{3.6}$$

Thus the function F defines one-to-one mapping between the sets {0,1,2, …, 383} and {-192, -191, -190,…,192} \ {0}. The equation 3.6 provides the values of carrier index k associated to index n that describes the relationships between inputs and the output of the frequency interleaver given in equation 3.4. The flow chart in Figure 3.8 illustrates the implemented frequency interleaving.

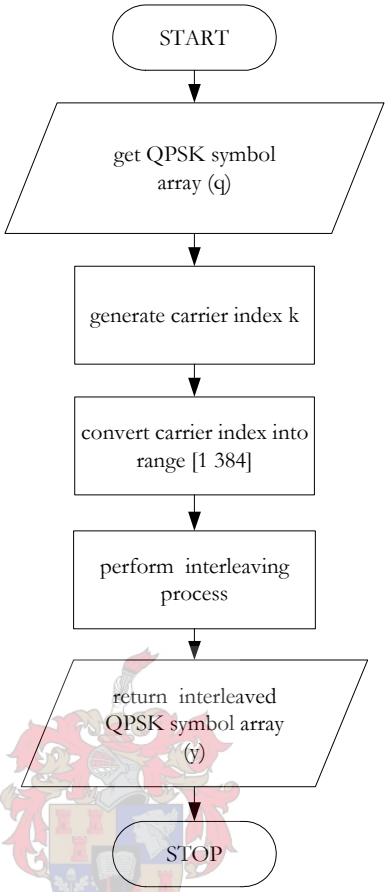


Figure 3.8 Frequency interleaving flow chart

The interleaving process starts with the generation of the carrier indexes k and ending with re-ordering of the QPSK symbols of the input array. The carrier indexes k are generated as shown in the following MATLAB code fragment:

```
TT(1)=0;            %initialise a permutation set of integers with TT(1)=0
F=[ ];              % define array  for  carrier index (k=F(n) function).
for i=2:512
   TT(i)= mod((13*TT(i-1) +127),512);

  % compute the carrier index  using elements
 % in the range[64 448] and excludes 256
  if ((TT(i)>=64) && (TT(i)~=256) &&(TT(i)<=448))
       F=[F  (TT(i)-256) ];
   end
end
```

The values of Π that lie in the range [64 448], omitting all others and 256 (the centre carrier), are selected. This yields 384 different values and 256 is subtracted from each value giving k values illustrated in Table 3.1

i	Π(i)=TT(i)	d_n	n	k
1	0			
2	127	127	1	-129
3	242	242	2	-14
4	201	201	3	-55
5	180	180	4	-76
6	419	419	5	163
7	454			
8	397	397	6	141
9	168	168	7	-88
10	263	263	8	7
11	474			
12	145	145	9	-111
13	476			
14	171	171	10	-85
15	302	302	11	46
16	469			
17	80	80	12	-176
18	143	143	13	-113
19	450			
. . .	. . .	. . .		
509	140	140	381	-116
510	411	411	382	155
511	350	350	383	94
512	69	69	384	-187

Table 3.1 The frequency-interleaving rule for the transmission mode II

MATLAB does not work with zero or negative indexes in the matrices. So we cannot work with the standard format of the values illustrated in Table 3.1. The solution is converting the carrier index in the range [-192 192] excluding zero, into positive carrier index range [1 384] where the central carrier is not used. The transformation is done with the addition of 193 to the negative carrier index and 192 to the positive carrier index. Thus for a carrier index k=-129, its

corresponding positive value is k=64, for k=-14, its corresponding positive value is k=179 and so on.

The frequency interleaving process operates according to the relationship defined in equation 3.4, where indexes k and n are obtained as illustrated in Table 3.1. The position of each value in series given in Table 3.1 provides the QPSK symbol index 1,2…384 and these values provides their correspondent carriers index k. Note, from Table 3.1 the generated carrier indexes (k) are in random order and the QPSK symbol indexes (n) are in a sequential order. According to the values in the table, when the QPSK symbol index n=1, the corresponding carrier index k= 64(-129), so when the interleaving process is applied a QPSK symbol of index n=1 in the array q, will map into index k=64 in array y, similarly the QPSK symbol of index n=2 in array q, will map into index k=179(-14) in array y, and so on. This changes the original order of the input QPSK symbols and results in a new random order in array y. That is frequency interleaving which combat the effect of frequency selective fading. The code fragment below shows how the interleaving process is carried out:

```
Y=zeros(1,(length(F)));   % initialise  an array Y with zeros equal to the
                          %carrier index
                          % Y holds the re-order qpsk symbol
 for v=1:length(F)
   if F(v)<0
      post_ind(v)=F(v) + 193;    % if the carrier index is negative add 193
      Y(post_ind(v))=q(v);       %map  qpsk symbol in q into Y
   else
      post_ind(v)=F(v) + 192;    %if  carrier index is positve shift it by adding 192
      Y(post_ind(v))=q(v);       %map qpsk symbol in q  into Y
   end
 end

mapqpsk=Y;                       % return the interleaved qpsk symbol
```

Figure 3.9 and Figure 3.10 illustrate the order of the QPSK symbols in the array before and after frequency interleaving respectively. In both figures S denotes the QPSK symbols.

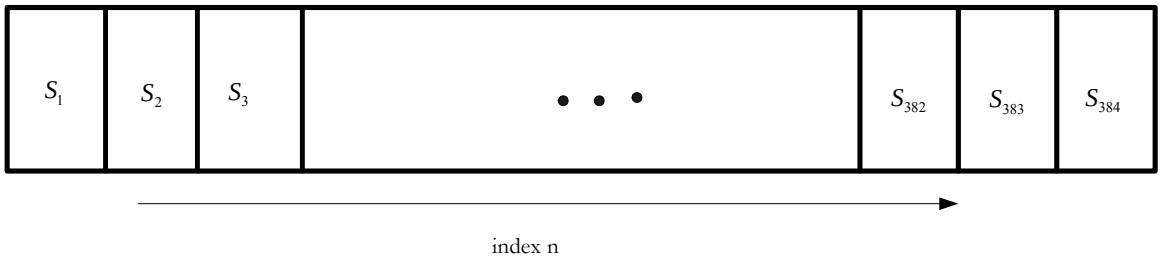


Figure 3.9 QPSK symbol array pre-frequency interleaving

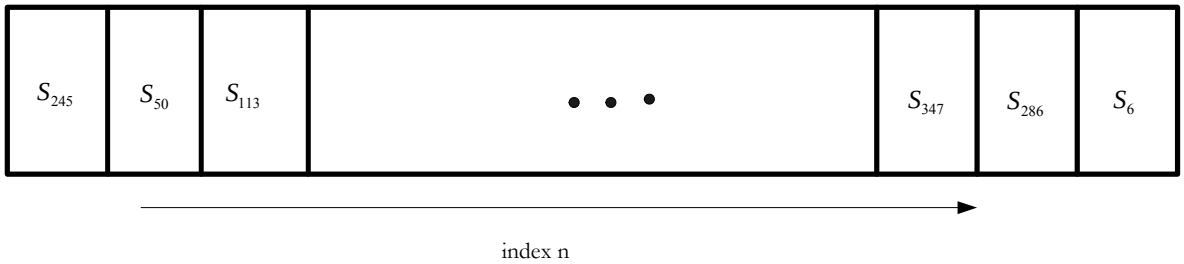


Figure 3.10 QPSK symbol array after frequency interleaving

3.5 Phase reference symbol generator

The DAB transmission frame described in section 2.6 follows a fixed format that allows receiver synchronization and extraction of data. The frame comprises of three channels namely the synchronization channel, FIC and MSC. The synchronization channel is made of the Null symbol and the Phase Reference symbol that provides receiver lock. This section describes the characteristics and the generation of the phase reference symbol.

According to [5] the first symbol of the transmission frame should be the Phase Reference symbol if a null symbol is not taken into account. The Phase Reference symbol appears only once in a frame. The receiver knows what the incoming Phase Reference symbol is supposed to be. The receiver gets the information about the behaviours of the channel in every single transmission frame by comparing the known pattern of the phase reference symbol with the received Phase Reference symbol.

The Phase Reference symbol provides the reference for the differential modulation of its neighbour OFDM symbol in the transmission frame. Its block has the capacity of 384 symbols (mode II), and the whole of this capacity is used for synchronization function. The pattern defining the phase reference symbol [5] is given next:

$$z_k = \begin{cases} e^{j\varphi_k} & for \quad \frac{-K}{2} \le k < 0 \quad and \quad 0 < k \le \frac{K}{2} \\ 0 & for \quad k = 0 \end{cases} \tag{3.7}$$

The value of φ_k will be obtained from the next expression:

$$\varphi_k = \frac{\pi}{2}.(h_{i,k-k'} + n) \tag{3.8}$$

where the indices i, k' and the parameter n are specified as the function of carrier index k (see Appendix B). The values of the parameter $h_{i,j}$ as a function of i and j are obtained from the table given in the Appendix B. The implemented flow chart that generates a phase reference symbol block is illustrated in Figure 3.11.

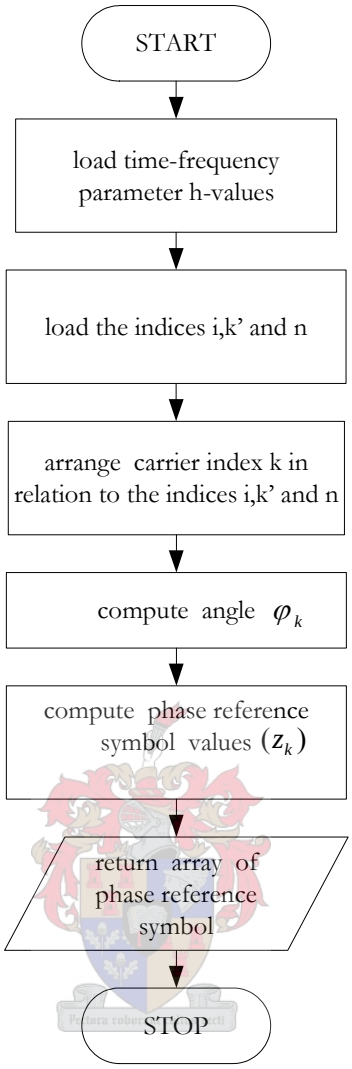


Figure 3.11 Phase Reference Symbol generation flow chart

The generation of the phase reference symbol follows a specific pattern described in the above paragraphs. The values of the parameter and indices for the transmission mode II are loaded according to the values given in Appendix B. The carrier index k is arranged in relation to the indices i, k' following the relation described in Appendix B. The Phase Reference symbol block is calculated according to equation 3.7. The expression (3.7) defines the value of z_k to be equal to zero at k = 0 in the carrier index range [-192 192], so that the central carrier is not used and the number of carrier elements in z is 384, for mode II. The carrier indexes k are set to positive values by adding 193 to the negative carrier indexes and 192 to the positive carrier indexes. This is done to ensure positive indexes when working with MATLAB.

The generated array of the phase reference symbol samples is used in differential modulation, as will be discussed in the next section. Figure 3.12 shows the generated phase reference symbol waveforms and Figure 3.13 shows its constellation. Its waveforms appear as the noise-like signal because the phases of the carriers are modulated in accordance with a predetermined rule.

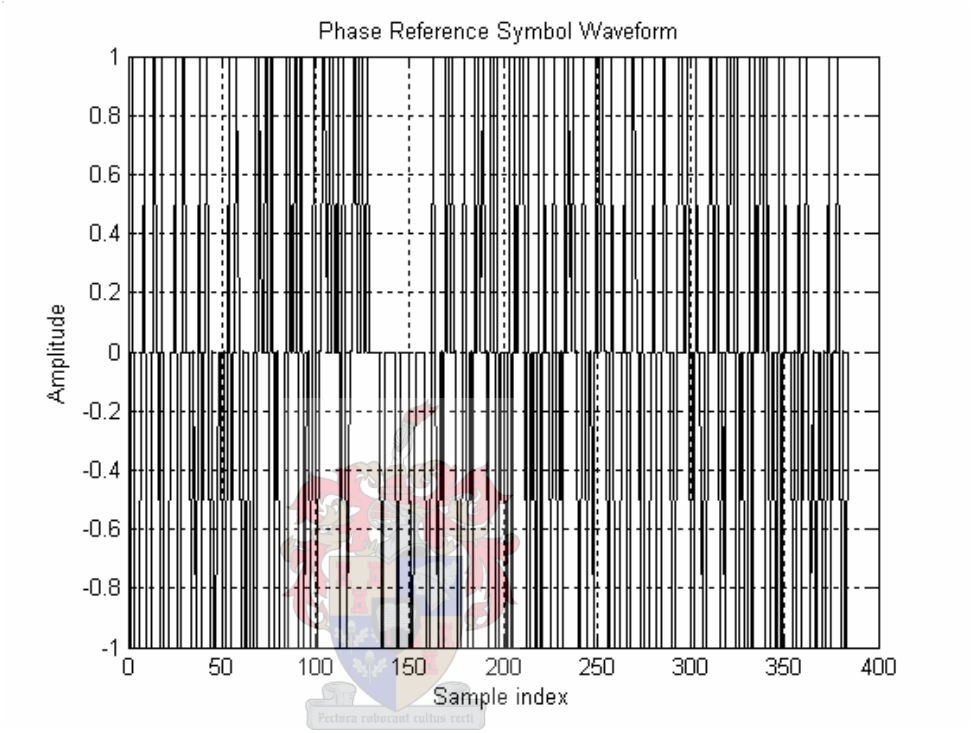


Figure 3.12 Real part of the phase reference symbol waveform.

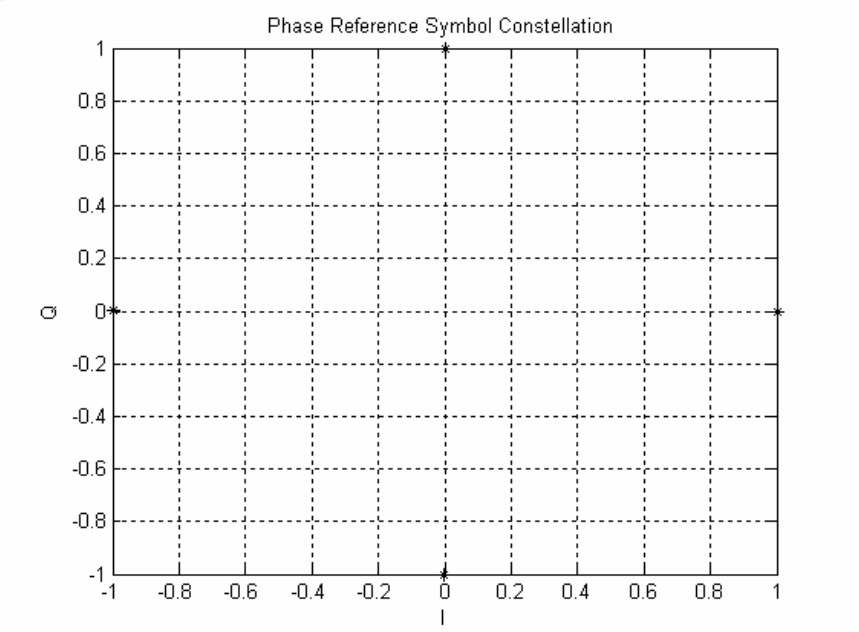


Figure 3.13 Phase reference symbol constellation.

3.6 Differential modulator

Up to this moment the information in the system is carried by the phase of the individual carriers (see Figure 3.7). The disadvantage of this transmission-reception is that the receiver has to be able to recognize the phase of the incoming carrier (coherent system). In mobile communications, the multipath spread of the signal can degrade or cause an offset in the phase of the carriers, so the solution is sending the information not through the absolute phase, but through the difference between the phases of two successive symbols. This simplifies synchronization and timing recovery, hence reduces receiver implementation and design complexity.

The differential modulator block is responsible for differentially modulating the QPSK carriers. The outputs of the block are the complex differential QPSK symbols that are used in the generation of the OFDM symbol for DAB signal as discussed in the second chapter. The $\pi/4$-shift D-QPSK modulation scheme is used. Recall the differential modulation works with a well-known phase reference. In this case the phase reference symbol generated in section 3.5

provides the initial phase reference (and a timing synchronization symbol) to the system as discussed in the second chapter.

The differential modulation standard [5] is defined by the following expression:

$$z_{l,k} = z_{l-1,k} \times y_{l,k} \tag{3.8}$$

$$for \quad l = 2,3...,L$$
$$and \quad -\tfrac{K}{2} \le k \le \tfrac{K}{2}$$

where z represents the complex differential symbol block, that is the output of the differential modulator, y represents the input QPSK symbol block, l represents the OFDM symbol index (see Figure 3.5) and k represents carrier index.

Expression 3.8 presents the complex differential multiplication of two arrays used in the simulation. The first array named "z" contains differential symbols and is also used to provide the phase reference information. The second array "y" has the QPSK symbols to be transmitted on each carrier. The two arrays have a similar capacity of 384 carriers (mode II). The elements of the array "y" are the output QPSK data stream from the data mapper presented in section 3.4. This output array is made of 75-QPSK-symbol blocks, each block with 384 QPSK carriers that constitute OFDM symbols of index l=2,3…76. In order to be able to transform the output "z" into input (z_{l-1}) according to equation 3.8, a feedback loop was necessary. Figure 3.14 illustrates the implemented detail for a differential modulator:

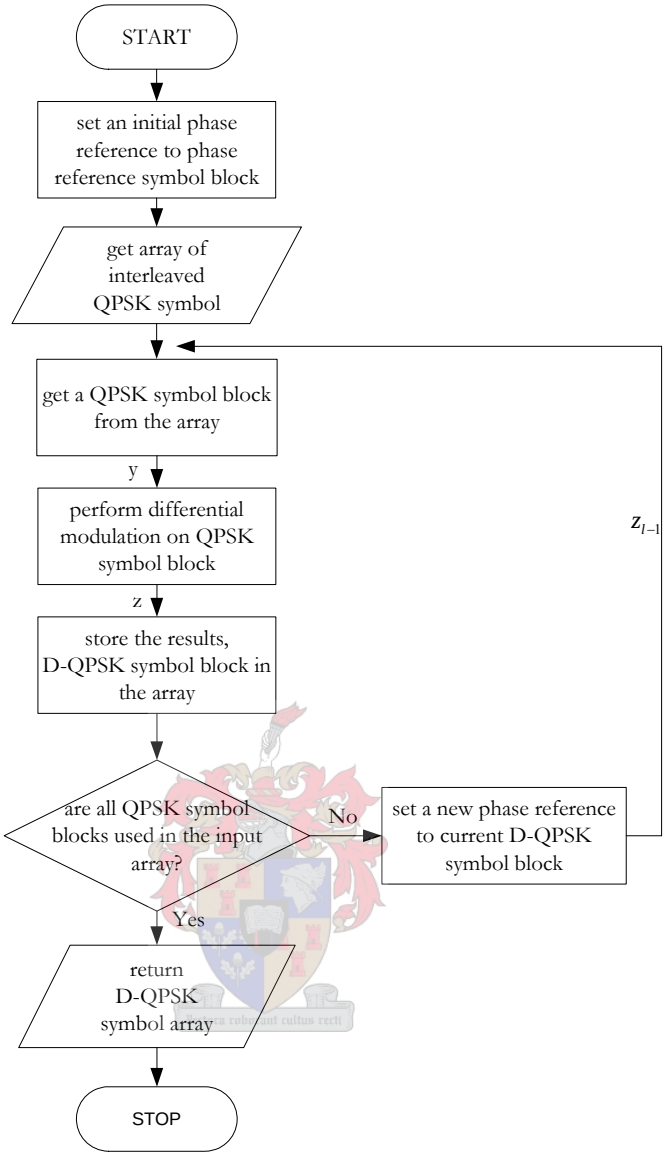


Figure 3.14 Differential modulator flow chart

From the input array a sequence of 384 QPSK symbols that form a QPSK symbol block are taken one at a time. The differential modulation is performed according to equation 3.8 on each QPSK symbol block. To get first DQPSK symbol block ($z_{2,k}$) of index l=2, the first incoming QPSK symbol block ($y_{1,k}$) from input array has to be multiplied by the phase reference symbol block. This first DQPSK symbol block obtained is then used to provide the phase reference for the second differential modulation. That is to get the second DQPSK symbol block ($z_{3,k}$) the first DQPSK symbol block has to be multiplied by the second QPSK symbol block ($y_{2,k}$) and

so on. The index l, of the DQPSK symbol block starts with l=2, since the first index is reserved for the phase reference symbol in the transmission frame.

When differential modulation is applied, the phase of each carrier is rotated by multiples of 45^0 from the previous OFDM symbol to the next. The four QPSK modulation states are signalled by $\pm 45^0$ and $\pm 135^0$ changes of the carrier phase at the start of each new OFDM symbol. The angle of the result of each carrier is the sum of the angles presented by the previous modulation and current QPSK carrier angle, and this defines the new DQPSK symbol. Thus the value of the new symbol determines the change of phase; hence the information on each carrier is carried by the phase difference. These features are illustrated in Figure 3.15 where the phase of one carrier is shown during three consecutive symbols; the actual phase shown is an example of many possible combinations.

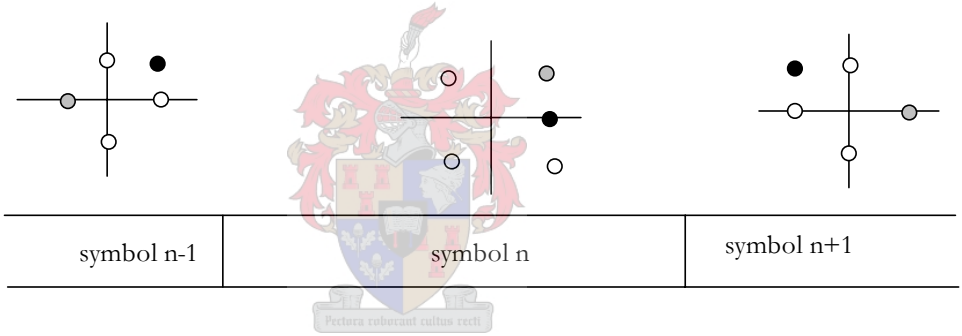


Figure 3.15 π/4-DQPSK modulation

3.7 OFDM symbol generator

As it was presented in chapter two, the OFDM symbol generator is the main block in the DAB transmission chain. It generates the OFDM symbols that make up the DAB main signal. This block transforms the frequency domain samples on every DQPSK symbol block into a time domain sample that presents the DAB main signal (see equation 2.8). The OFDM scheme has

already been presented in chapter 2, this section discusses how the OFDM symbols are generated in the simulation.

The output DQPSK data streams from the differential modulator block provide inputs to this block (see Figure 3.2). This input array is made of 75-DQPSK-symbol block each with 384-differential modulated carriers (mode II). In order to achieve the total number of the OFDM symbols desired for a transmission frame as described in chapter two, this block adds a phase reference symbol at the beginning of the input array, which results in the 76-symbol blocks array present in the block. This constitutes one transmission frame as illustrated in Figure 3.16.

1	2	3	….	…	75	76

Figure 3.16 Arranged symbol block in transmission fame

Note: The first index in Figure 3.16 presents the Phase Reference Symbol block and the rest are for FIC and MSC.

In order to ensure working with one DQPSK symbol block from the input DQPSK data stream at a time, the loop illustrated in Figure 3.17 was implemented. This is done because the generation of an OFDM symbol uses a single DQPSK symbol block, and the complete transmission frame is made of numbers of OFDM symbols that are generated with different DQPSK symbol blocks according to equation 2.8 presented in the second chapter.

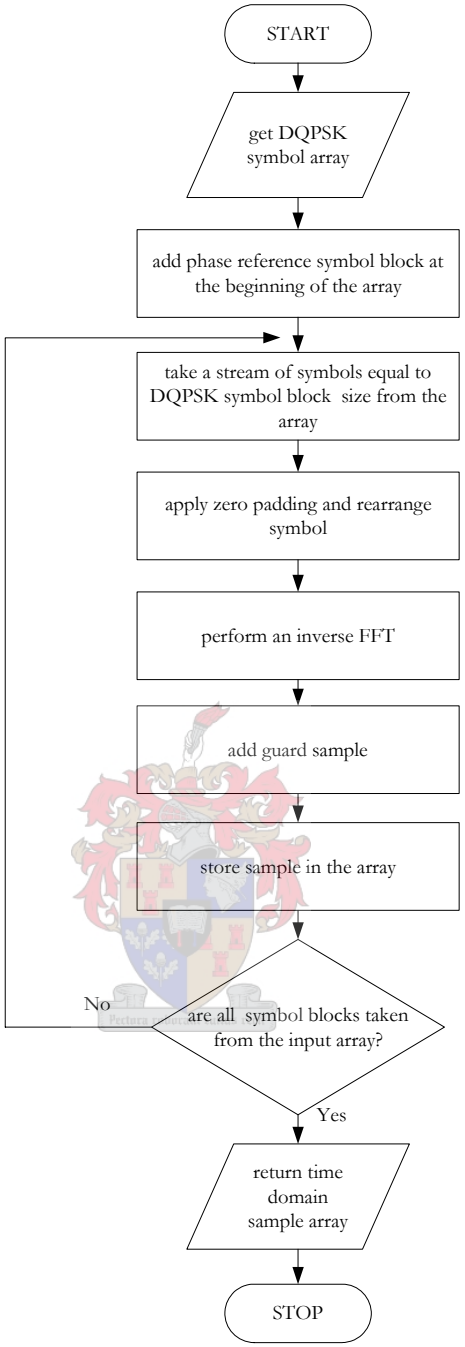


Figure 3.17 OFDM symbol generator flow chart.

The incoming DQPSK data streams forms 75 DQPSK symbol blocks that are added to the phase reference symbol block to form an array made of 76 symbol blocks. Each block consists of 384 symbols. As presented in the above paragraph one symbol block is used at a time starting with the phase reference symbol block that is at the beginning of the array. To generate a time

domain OFDM symbol from each symbol block, the following are to be performed on each symbol block:

- Zero padding
- Inverse FFT
- Guard sample insertion

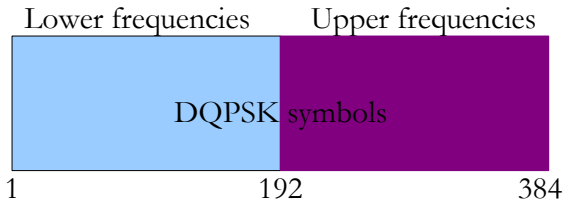
In order for the OFDM symbol generator to accomplish the above task, the block has to be divided into three sub-blocks. These sub-blocks are zero padding, IFFT and cyclic prefix, where each block performs its own function presented in sub-section 3.7.1, 3.7.2 and 3.7.3 respectively. In the following subsections a stream of 384 symbols will be denoted as D-QPSK symbol block. But it should be clear as presented in the above paragraph; the incoming DQPSK symbol array has concatenated to the phase reference symbol made of 384 symbol samples at the beginning of the array (see Figure 3.16 and 3.17).

3.7.1 Zero padding

The heart of the OFDM Symbol Generator is an inverse FFT algorithm. As was presented in chapter two, the FFT/IFFT algorithms have a good performance if the number of carriers is an integer power of 2 (see section 2.7). Since the length of the D-QPDK symbol block applied to the algorithm is not equal to a power of 2, zero padding is needed to fit this length equal to a power of two (i.e. from 384 to 512). This is one of the functions of this sub-block.

According to [5], the IFFT in DAB uses 512 samples for Mode II to produce a 384-carrier DAB symbol signal. To work with a 512-sample block, this sub-block has to add 128-zeros to every D-QPSK symbol block as illustrated in Figure 3.18

The mathematical representation of the DAB main signal in equation 2.8 uses both positive and negative frequencies. To ensure working with the same representations, the lower frequency on the first part of each OFDM symbol and the upper frequency in the last part of the OFDM, the samples from each D-QPSK symbol block are to be rearranged as illustrated in Figure 3.18 before an inverse FFT is performed. This has to be done because the FFT/IFFT algorithm changes the position of the carriers.



DQPSK symbol block before zero padding and rearrangement

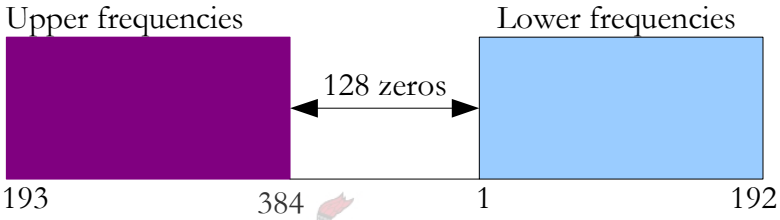


Figure 3.18 DQPSK symbol block after zero padding and rearrangement.

The samples presenting negative frequencies (-4kHz to -768 kHz) in the base-band signals are named as Lower frequencies and the samples presenting the positive frequencies (4kHz to 768 kHz) are named as Upper frequencies.

3.7.2 IFFT

This is the main sub-block of the OFDM Symbol Generator and the heart of the simulation. At this moment the system generates the OFDM symbols that present DAB main signal (see equation 2.8 in chapter 2). The frequency domain samples in each D-QPSK symbol block are transformed into time domain samples. To accomplish this, the sub-block performs an inverse IFFT on every D-QPSK symbol block with the desired shape obtained from sub-section 3.7.1. The carriers in the symbol block are modulated into orthogonal carriers that form OFDM symbols. The output of the block is the OFDM symbols. No MATLAB code has been written for this sub-block; the built in IFFT algorithm has been used with 512-IFFT points (mode II).

3.7.3 Cyclic prefix

This sub-block creates cyclic prefix as presented in section 2.7 in the second chapter. It takes a copy of 126 (mode II) samples corresponding to the guard interval period (see Table 2.1) from the end of every OFDM symbol and then copies them at the beginning of the symbol. This extends the length of the OFDM symbol to 638 samples length (see Table 2.1).

3.8 Null symbol generator

This is the last block in the transmission side. The block restructures the output of the OFDM symbol generator into a form desired for transmission. It performs the two functions described next:

- Null symbol generation
- Final frame structure formation.

The output of this block is the complex base-band signal presenting the DAB main signal s(t) described in the second chapter. Figure 3.19 illustrates a complex base-band signal generated in the simulation made of one transmission frame. The complex base-band signal generated comprises of 384 orthogonal carriers generated in section 3.7 with equal carrier spacing.

3.8.1 Null symbol generation

The first OFDM symbol of transmission frame is the Null Symbol with the duration T_{NULL} (see section 2.8). The T_{NULL} is equivalent to 664 samples in Mode II (see Table 2.1). To ensure that no main signal is transmitted during this period, 664-zeros are generated to present the null symbol. The MATLAB code generating the null symbol is shown below:

null_symb=zeros (1,664) (generating a Null Symbol).

3.8.2 Final frame structure formation

A complete transmission frame must have a Null Symbol at the beginning of the frame. The time domain sample array obtained from the OFDM Symbol generator constitutes samples for one transmission frame without a null symbol. To have a complete transmission frame a null symbol has to be added at the beginning of this time domain sample array.

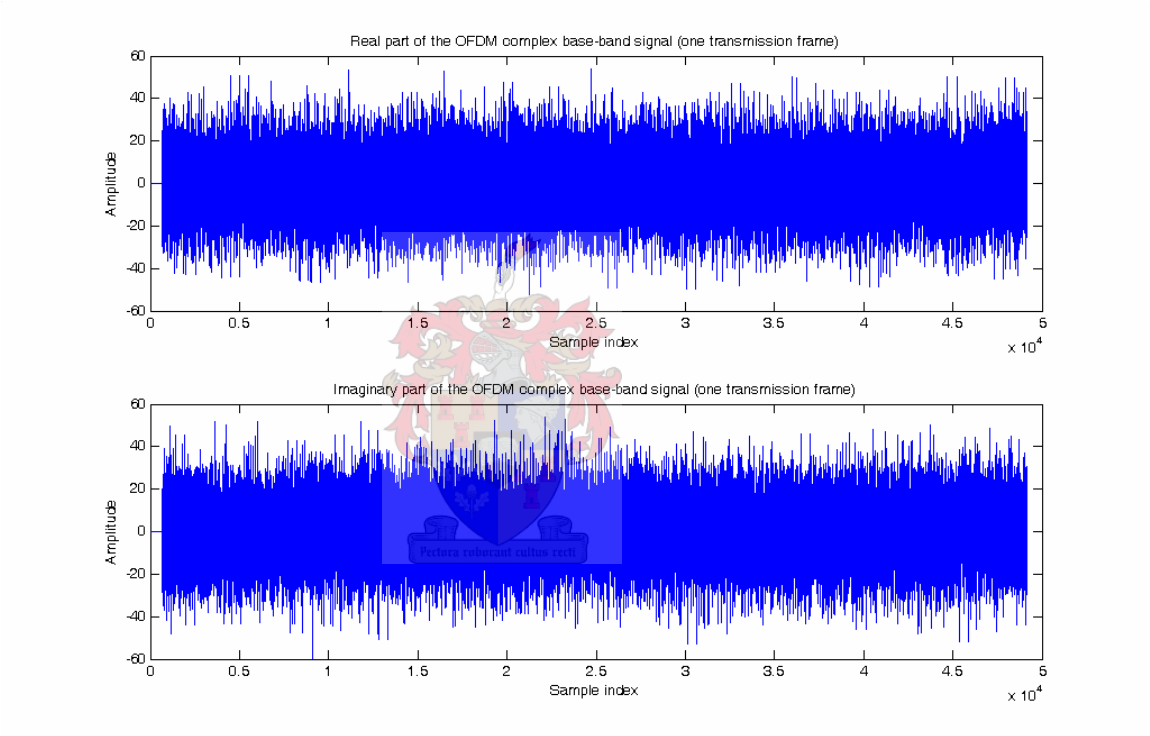


Figure 3.19 Generated complex base-band DAB signal.

3.9 Channel

A fixed receiver communication has been simulated and the channel used is an additive white Gaussian noise channel. The complex white Gaussian noise was generated and added to the complex base-band signal in the simulation. The MATLAB function 'randn' was used to generate this noise where the real and imaginary components were uncorrelated and both treated with equal noise amplitude level.

3.10 Reception side

The DAB standard [5] provides only the transmission system standard. Nothing is said in the standard about how to design the DAB receiver. This is because the reception system should be open to promote the competition among receiver designers and manufacturers, but should ensure that all the receivers are able to work with DAB signal.

The simulation model for the DAB system has been completed with the basic design of a DAB receiver. A basic receiver design follows the inverse of the transmission process. Its design approach is illustrated in Figure 3.2 and the detail of each block is provided in next sections.

3.11 Synchronization

Synchronization in different layers is a challenging but very important issue in a digital communication system. Frame synchronization, carrier synchronization, and symbol timing synchronization in the physical layer are usually the most important. Frequency offset and symbol timing error in a receiver are the most often encountered problems in a digital communication system.

The receiver simulation does not include the RF section neither analogue to digital conversions. The simulation has been done in a base-band format. The received data signals are stored in the array. From the array the synchronization is acquired to provide symbol timing and frame timing before demodulation.

The symbol timing synchronization and the frame synchronization has been implemented fully in the receiver simulation model. Unfortunately the complete implementation for compensating the frequency offset (i.e. controlling the RF local oscillator) in the receiver simulation was not implemented since the ADC and the down converter part were not studied in this thesis. It was assumed to be a perfect working hardware. But explanations on how it should be done if a complete system simulation is considered, will be introduced in section 3.13.

The frame synchronization is used for rough estimates of the frame timing. It exploits the presence of the null symbol in the transmission frame. The symbol timing synchronization is used to reduce the residual symbol timing error of the frame synchronization. The details for symbol timing synchronization and frame synchronization are given in section 3.12.1 and 3.12.2 respectively. Figure 3.20 illustrates the receiver synchronization process for a complete system.

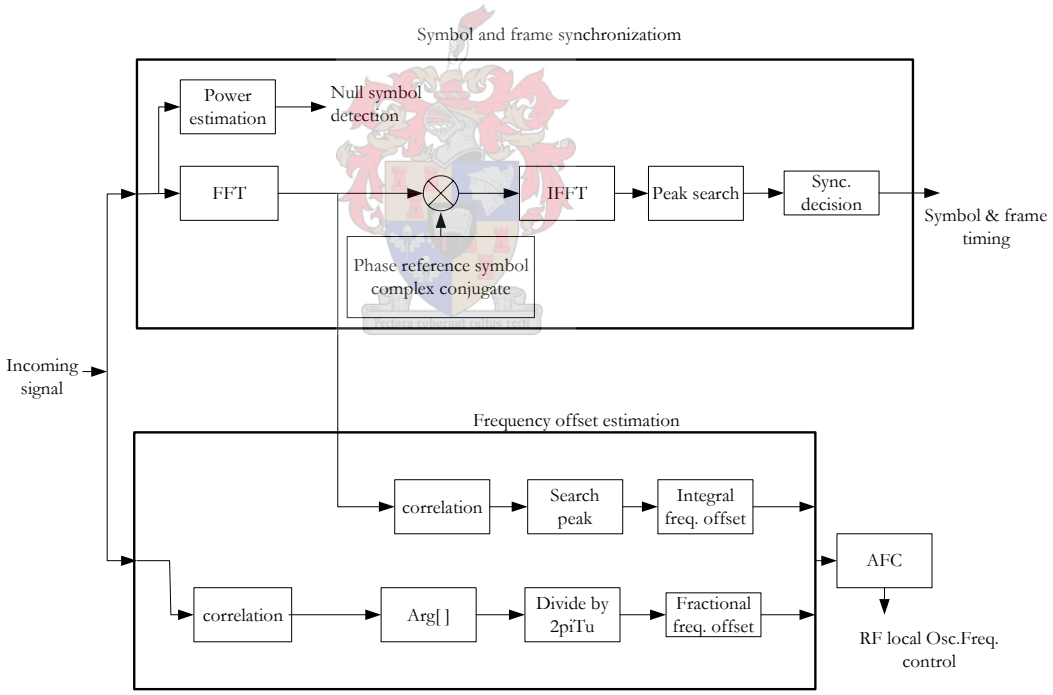


Figure 3.20 Block diagram of synchronization process

Below is the synchronization procedure when data signal is received:

- Determine symbol timing, simultaneously detect the occurrence of null symbol (Frame timing).
- Estimate and compensate for the fractional frequency offset
- Estimate and compensate for the integral frequency offset

3.12 Timing synchronization

The timing synchronization estimates the start of the frame and provides the correct symbol timing. To ensure these are achieved, symbol timing synchronization and frame synchronization are implemented. The next two sections discuss the details of their implementations.

3.12.1 Symbol timing synchronization

The symbol timing synchronization estimates and finds the start of the phase reference symbol in the received data. This provides the correct timing within the symbol period to take the received data samples. It uses the received phase reference symbol to measure the impulse response of the transmission channel, which provides accurate symbol timing and frame timing.

For clear understanding of how the received phase reference is used to measure the channel impulse response (CIR) and to provide accurate timing, let's consider the information provided in the following paragraphs:

Consider a timing error within guard interval. This results into a linear phase shift of each sub-carrier in frequency domains as illustrated in equation 3.9.

$$Z_r(k) = Z(k)e^{-j2\pi kn_0/N} \tag{3.9}$$

Where Z (k) is the k-th sub-carrier of the phase reference symbol in frequency domain, Z_r (k) is the k-th sub-carrier of received phase reference symbol with a time delay of n_0 (i.e. demodulated by an FFT with n_0 offset) and N is the number of sub-carriers.

The phase reference symbol is the dedicated pilot symbol in DAB transmission frame [5]. When a received phase reference symbol is multiplied by the complex conjugate of the phase reference symbol spectrum at the receiver, this results in a modulated phase of each carrier being eliminated. The resultant product is the phase-demodulated received phase reference symbol, which contains the time delay (n_o) information.

The CIR, h(n) is estimated by performing an inverse FFT of the resultant product as shown in the next expression:

$$h(n) = IFFT\{Z^*(k) \bullet Z_r(k)\} = \delta(n - n_0) \tag{3.10}$$

where $Z^*(k)$ is the complex conjugate of Z (k), and $\bullet$ denotes the element-by-element product of vectors and n is the sample index in time domain. The peak of CIR indicates start of the phase reference signal.

The above paragraphs have illustrated how the CIR is measured, Figure 3.21 illustrates the implemented symbol timing synchronization loop.

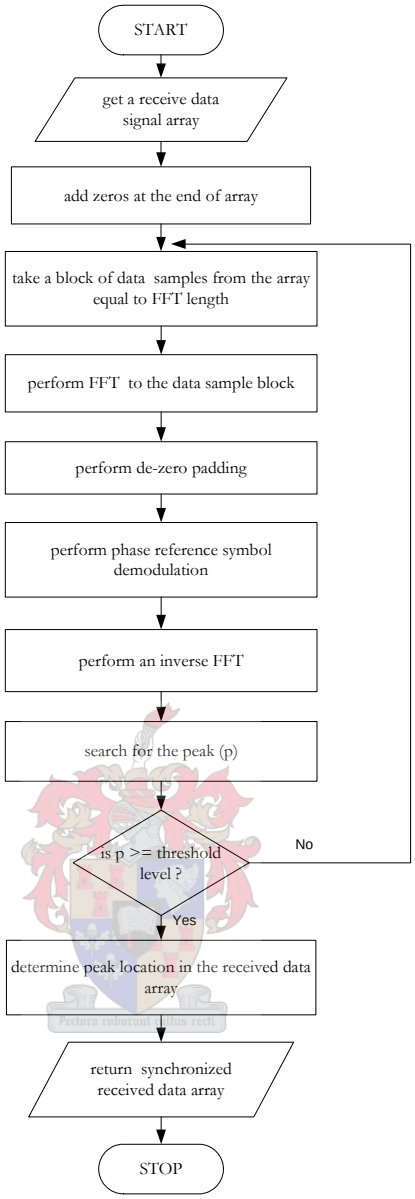


Figure 3.21 Symbol timing synchronization flow chart

As was explained earlier, the received data signal is stored in the array. The array size is not necessarily a multiple of FFT length, where a sample-block with size equal to FFT length is taken. To facilitate working with the array in MATLAB, extra zeros have to be added at the end of the received data signal. This results in an array multiple of FFT length.

The symbol timing synchronization is accomplished through two stages, the first stage measures the CIR using the phase reference symbol from the received data array, and the second stage determines the start of the phase reference symbol. The detail of each stage is explained next.

1) Determining of CIR using phase reference symbol from the received data signal

From the received data array, a data sample block equal to the FFT length is taken. The samples in the block are transformed into frequency domain by performing the FFT operation. The output frequency domain samples contain frequency response information about the channel.

The FFT output array has a size equal to FFT window length, but the size of phase reference symbol block at the receiver is 384 (mode II). So zero padding removal and data rearrangement has to be done to reverse the process done at the transmitter side. The obtained array after zero padding removal has the same size as the known phase reference symbol block at the receiver. When this array is multiplied by the complex conjugate of the receiver phase reference symbol block, this results in the array with information about the frequency response of the channel, which is then related to the (time) impulse by performing an inverse FFT operation to the product. The IFFT output array provides a measure for channel impulse response (CIR).

2) Determining of the start of the phase reference symbol in the received data array

The impulse signal is determined by calculating the magnitude of each element in the IFFT output array and searching for the desired highest peak. It does not mean that for every sample block taken from the received data array, the desired peak that determines the start of the phase reference symbol will be found. To ensure a desired peak is determined a threshold level has to be set, as it will be explained later.

For every output array of an inverse FFT corresponding to the sample block taken from the received data array (see Figure 3.21), the highest peak has to be determined and compared to the set threshold level. When the determined highest peak is less than the threshold level, then the peak found does not indicate the start of the phase reference symbol, so the loop process continues by taking the next sample block (see Figure 3.21).

The determined peak is only greater than the threshold level provided the used data sample block has a phase reference symbol sample in it, since the phase reference symbol has a high correlation with itself. So when a peak found is greater than the threshold level, this peak indicates that the samples from the useful part of the phase reference symbol were present in the used sample block. To determine the start of the phase reference symbol in the received data array, the location of the peak has to be determined.

The numbers of loop trials (N_l) before determining the desired peak has to be recorded. When the desired highest peak is found, the start of the phase reference symbol in the received data array is given by expression below:

$$prs_{start} = fftlength \times N_l + peak_{location} \tag{3.11}$$

where

prs_{start} denotes the start of the phase reference symbol in the received data array.

$fftlength$ is the size of FFT window used.

$peak_{location}$ is the location of the peak in IFFT output array.

The start of the phase reference symbol in equation 3.11 indicates the starting point of the useful symbol duration for the phase reference symbol. To determine where to start demodulating the received OFDM symbol in the array, the FFT length has to be added in equation 3.11. This is because the received phase reference symbol was added to zeros in order to fit to the FFT length at the transmitter. The start of the useful symbol part of the phase reference symbol in the received data array is illustrated in Figure 3.22.

Figure 3.22 Start of effective phase reference symbol

The simulated highest peak location is illustrated in Figure 3.23. The plot in the figure has been obtained with one transmission frame made of three OFDM symbols.

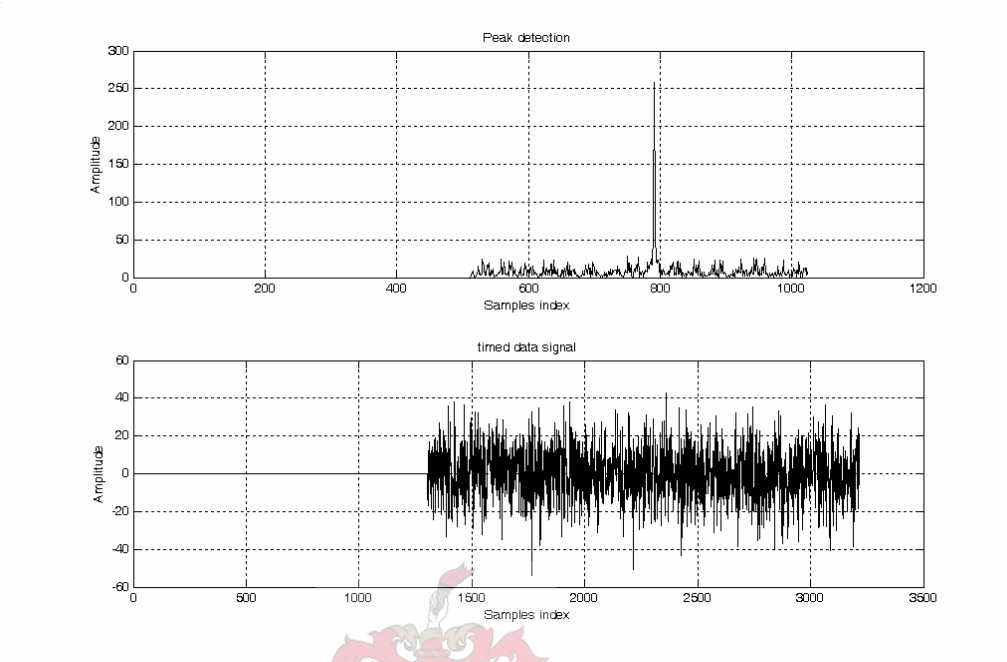


Figure 3.23 Symbol and Frame timing synchronization.

From Figure 3.23 the peak is at sample index 791. The plot includes the entire received data sample before and during desired peak detection. It is well known from chapter 2 that the first symbol in the transmission frame is the null symbol of sample length 664 and each OFDM symbol has the cyclic prefix sample of 126 (transmission Mode II) at the beginning of the symbol. The useful symbol duration does not include the cyclic prefix (i.e. guard interval sample).

Adding the null symbol samples and the cyclic prefix samples we obtain 790, thus the peak in the figure appears precisely at the starting point of the useful OFDM symbol duration (Tu) of the phase reference symbol as expected. The timed received data is also shown in the same figure, with a real part of the received data signal plot.

To set the threshold, the magnitude of the highest peak when the phase reference symbol is multiplied by its complex conjugate and an inverse FFT is applied to the products in the absence of the noise and in the presence of the noise has to be observed. Also a similar

observation has to be done using the incoming signal. The incoming signal in the presence and absence of noise is multiplied by the complex conjugate of the phase reference symbol and an inverse FFT is applied to the product. The magnitude of the highest peak from IFFT output array in both cases has to be observed. These are done to ensure that during timing process the noise peak will not be considered as the desired peak as well as the incoming signal peak excluding the phase reference symbol signal. It was concluded that the threshold level is to be greater than half the magnitude of the peak in the absence of the noise (see Figure 3.24). This provided a better result under either case; the performance results will be discussed in details in section 3.17.

The Figure 3.24 shows the peak when the phase reference symbol is correlated by itself and then performs the IFFT operation.

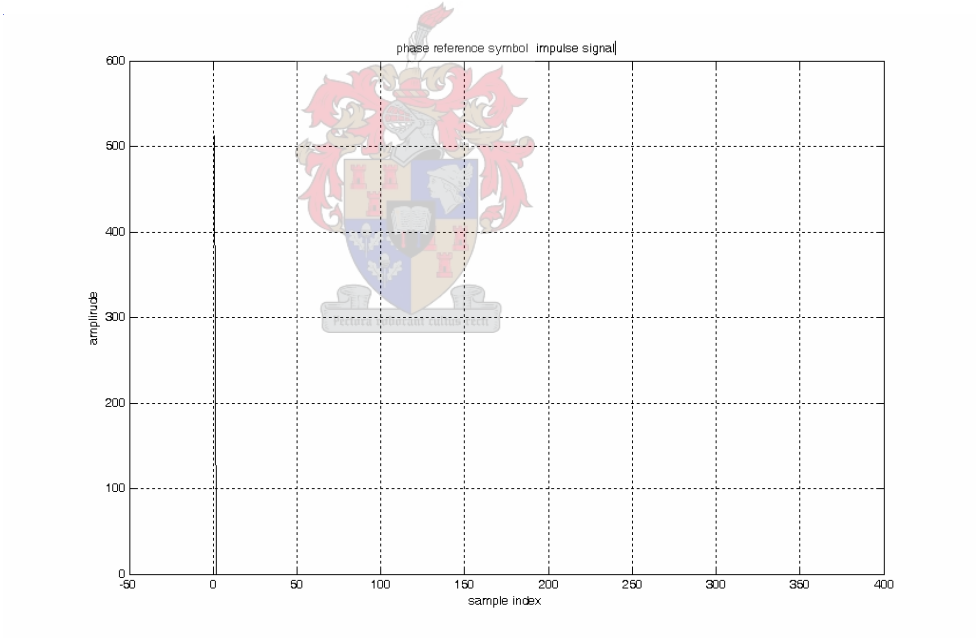


Figure 3.24 Phase reference symbol impulse signal

3.12.2 Frame synchronization

The Null Symbol is the first symbol in the DAB transmission frame and no signal is transmitted during the null symbol period. The frame timing is roughly estimated using null symbol detection by measuring the signal average power during the null symbol period. From the incoming signal, the samples equivalent to the size of the null symbol period are used to measure the average signal power. When average signal power is less than half of the average transmitted signal power the null symbol has been detected, hence the start of a new frame. But this does not guarantee accurate frame timing because it does not work well in low SNR environment; instead the symbol timing synchronization described in above section is to be used to provide correct timing. The detection of the phase reference symbol also indicates a new frame since the phase reference symbol occurs once in each transmission frame.

3.13 Frequency offset estimation and correction

In OFDM systems the sub-carriers are exactly orthogonal only if the transmitter and the receiver use exactly the same frequencies. Thus the receiver has to estimate and correct the carrier frequency offset of the received signal. In the DAB system, the correction for the carrier frequency offset is done through use of an automatic frequency control (AFC) signal [34]. This signal is used to digitally adjust the IF Oscillator (RF Local Oscillator frequency) (see Figure 3.20). The AFC constitutes of two components:

- Estimated Fractional Frequency offset
- Estimated Integral frequency offset

The operation starts with determining the fractional frequency offset and compensates for it. It is then followed by estimating the integral frequency offset. The estimated values are converted into a voltage that is used to control the local oscillator frequency and clock [13]. The detail of the estimates for each component follows in sub-section 3.13.1 and 3.13.2 respectively.

3.13.1 Fraction frequency offset estimation

An OFDM symbol is preceded by a cyclic extension that is the copy of the last portion of the symbol. The first Tg seconds of the OFDM symbol is identical to the last part. This property can be exploited to estimate frequency offset less than one of the carriers' spacing (a fraction of frequency offset Δf_{fr}) using the scheme as depicted in Figure 3.25.

Consider when a received signal r(t) reaches the last Tg (guard interval) period of a symbol(see Figure 3.25) and assumes a frequency offset given in equation 3.12 exists.

$$\Delta f = \Delta f_{fr} + \Delta f_{in} \tag{3.12}$$

where Δf_{fr} is the real number with absolute value less than one and Δf_{in} is an integer.

Let's denote the output of the correlator as z(t). When the received signal r(t) is correlated with a version time shifted by the useful part of symbol duration Tu, the output of the correlator is given by :

$$\begin{aligned} z(t) &= r(t)r^*(t-Tu) \\ &= a(t)\exp\{j2\pi(\Delta f_{fr}+\Delta f_{in})t\} \\ &\quad a^*(t-Tu)\exp\{-j2\pi(\Delta f_{fr}+\Delta f_{in})(t-Tu)\} \\ &= C(t)\exp\{j2\pi\Delta f_{fr}Tu\} \end{aligned} \tag{3.13}$$

Where $a(t)$ is received symbol when frequency offset is zero and

$$C(t) \equiv a(t)a^*(t-Tu) \approx a(t)a^*(t) = |a(t)|^2 > 0$$

The fractional frequency offset can be estimated from:

$$\Delta f_{fr} = \frac{Arg[z]}{2\pi Tu} \tag{3.14}$$

Figure 3.25 shows the point-to-point correlation summed over guard interval.

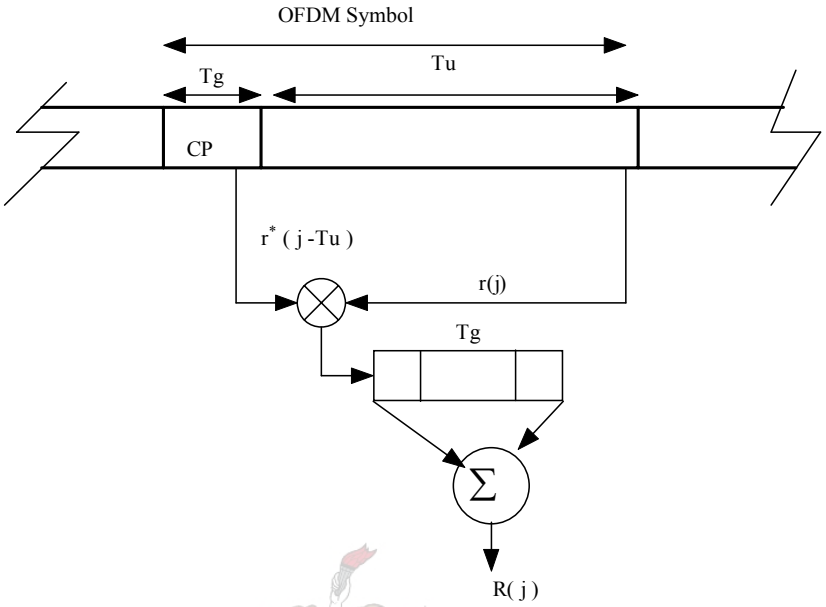


Figure 3:25 Point-to-point correlation

This scheme correlates Tg second of the OFDM symbol with a part that is Tu seconds delayed (Tu being the useful symbol duration). The output of the correlator is written as:

$$R(j) = \sum_{n=1}^{Tg} r(j-n)r^*(j-n-Tu) \tag{3.15}$$

The cross correlation is performed as shown in equation 3.15 and given in Figure 3.25 for each symbol and summed over guard interval sample. The output of the correlation for each symbol is stored in the array R. After OFDM symbol correlation, the absolute value of the elements in the array R is computed. Then the maximum correlation peak is searched and its location is determined. The location of the highest peak provides the sample index (j) that provides the value of R used in the computation of the correlation phase.

The fraction frequency offset is estimated from the maximum correlation peak at the output of the correlator. The phase of the correlator output is equal to the phase drift between samples

that are Tu seconds apart. The fractional frequency offset is estimated from the correlation phase divided by $2\pi Tu$ (see equation 3.14).

3.13.2 Integral frequency offset estimation

The integral frequency offset (Δf_{in}) is estimated using the phase reference symbol. A received phase reference symbol with frequency offset in the frequency domain (after FFT operation) is cross-correlated to a series of known receiver phase reference symbol at the receiver. The location of the highest output peak giving a maximum correlation has to be determined and from this location the integral frequency offset is determined. The highest peak is basically expected to occur at the origin if there is no frequency offset, thus the amount of points that the highest peak shifts provides the integral frequency offset.

3.14 OFDM symbol demodulator

After symbol timing synchronization has been accomplished at the receiver, the output array of the synchronization block contains a data signal made of OFDM symbol samples that exclude the null symbol and the phase reference symbol guard interval samples. The OFDM demodulator block is responsible for the demodulating of the OFDM symbols from synchronized data array. It is the main block in the receiver side. The output of the block is the DQPSK symbol blocks made of DQPSK symbols placed in the same order as the output signal of the differential modulator.

The block consists of three sub-blocks shown below that works together to accomplish the block responsibility. The details of these sub-blocks are given in the sections 3.14.1, 3.14.2, and 3.14.3 respectively. The three sub-blocks are:

- Cyclic prefix removal
- FFT
- Zero padding removal

3.14.1 Cyclic prefix removal

This is the first sub-block of the OFDM symbol modulator block. The block removes the guard interval samples added to each OFDM symbol at the transmitter. Its output is the OFDM symbol blocks of the useful part of the OFDM symbol period. This provides inputs to the FFT block. Remember the guard interval samples for the phase reference symbol have been removed in the synchronization block. The first 512 samples from the input array are fed directly to the FFT block since they belong to the phase reference symbol.

3.14.2 FFT

Every OFDM symbol block after cyclic prefix removal has a size equal to FFT length, which is a power of two. The FFT block performs the FFT operation to every OFDM symbol block. This transforms the OFDM symbols back to the frequency domain. There is no MATLAB code written for this block, only the built "fft' function in MATLAB has been used.

3.14.3 Zero padding removal

This is the last sub-block of the OFDM symbol demodulator. The block removes the zero padding and rearranges the data in a suitable form to feed the differential demodulator. This ensures the output of the OFDM symbol demodulator with the differential QPSK symbols placed in the same order as the output signal of the differential modulator. So the lower indices are replaced at the beginning of the DQPSK symbol block and the upper indices at the last part of the DQPSK symbol block. The process involved in the zero padding removal and data rearrangement is illustrated in Figure 3.26.

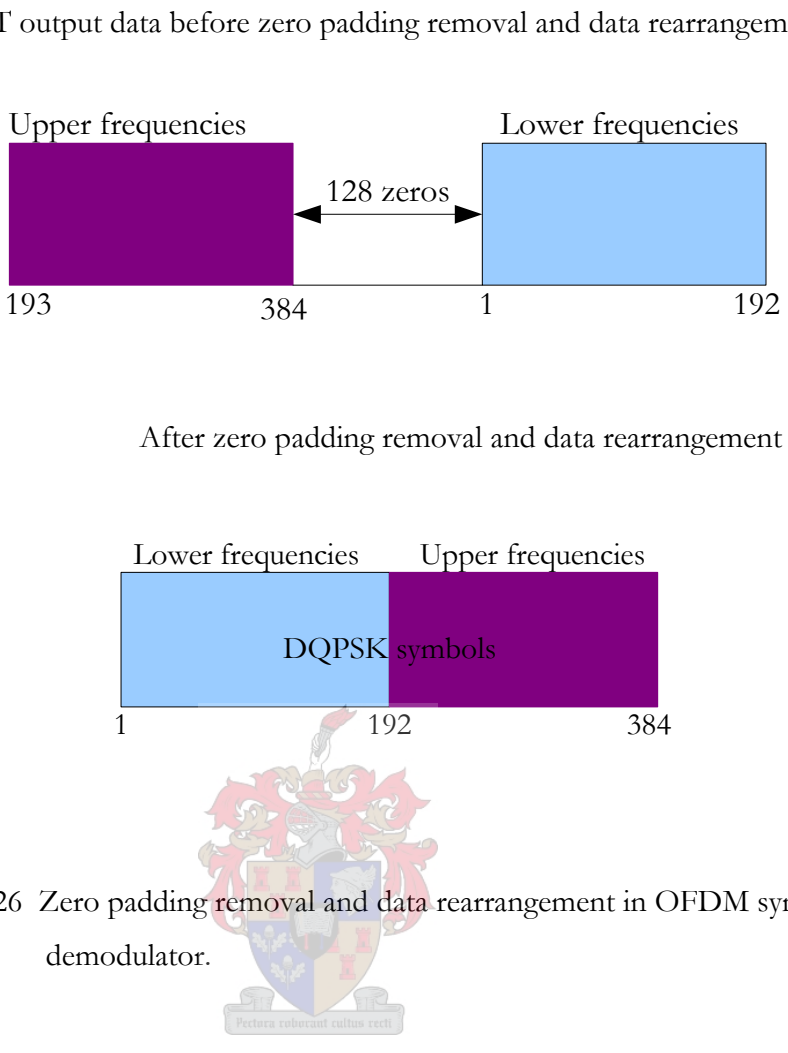


Figure 3.26 Zero padding removal and data rearrangement in OFDM symbol demodulator.

3.15 Differential demodulator

After OFDM symbol demodulation has been performed as explained in the above section, the desired DQPSK symbol blocks are obtained. The differential demodulator performs the DQPSK demodulation to every DQPSK symbol block using a complex differential multiplication defined in the next expression:

$$y_{l,k} = z_{l,k} * z^{*}_{l-1,k} \tag{3.16}$$

Where "y" represents output of the differential demodulation operation, "z" presents the input DQPSK symbol block, "z*" presents the complex conjugate of the differential phase reference, where the initial phase reference is provided by the received phase reference symbol block and the rest is obtained from the previous DQPSK symbol block. l and k are the OFDM symbol

index and carrier index respectively. The output of the block is the QPSK symbols that provide input to the data de-mapper block.

3.16 Data de-mapper

This is the last block in the receiver side. It transforms the QPSK symbol into bits, so its output is the original information as the input of the transmitter. This block is made of the two main sub-blocks given below and their details follow in the next subsections.

- Frequency de-interleaving
- QPSK symbol de-mapping

3.16.1 Frequency de-interleaving

This block performs the inverse of the frequency interleaving presented in section 3.4.3. The carrier indexes are rearranged again over the QPSK symbol index using the inverse of the rearrangements done in the frequency interleaving. The outputs are QPSK symbols arranged in a similar way as the output of the QPSK symbol mapper at the transmitter side. Its output provides input to the QPSK symbol de-mapper block.

3.16.2 QPSK symbol de-mapper

This is the last sub-block of the data de-mapper. It is responsible for the transforming of the complex QPSK symbols at the output of the frequency de-interleaving again into the bits stream. Thus the original information is recovered. Its operation is the reverse of the QPSK mapper presented in section 3.4.2.

From the QPSK mapping constellation in Figure 3.6, a symbol (01) has positive real part and a negative imaginary part. So when a sign of the real part of the complex QPSK symbol is positive, the decoded bit is "0' and when it is negative the decoded bit is "1". This applies similarly to the imaginary part of the complex QPSK symbol.

The bits from each QPSK symbol block are rearranged in order according to how the bit pairs was used in the formation of the QPSK symbol in the QPSK symbol mapper block. For example when a first symbol in the QPSK symbol block is decoded, the decoded I-phase component bit will be directed to the first index in the bit stream block and its counter part Quadrature components will be directed to the 385 index and so on. This has to be done to ensure the de-mapped bit streams are in the correct order.

3.17 Results and Conclusion

The aim of the simulation of the DAB model system has been achieved, since the transmitted data has been correctly recovered at the receiver. Three OFDM symbol messages of 2304 bits length have been generated at the transmitter input and regenerated at the receiver output. But in order to realise the performance of the simulated system model and how the model works, the system symbol timing synchronization and bit error rate performance were analysed. The results of the performance are presented next.

a) Symbol timing synchronization

The system symbol timing synchronization has been investigated in better and worse signal to noise ratio environments. The plot in Figure 3.27 shows the system symbol timing synchronization performance with SNR of –9.85dB.

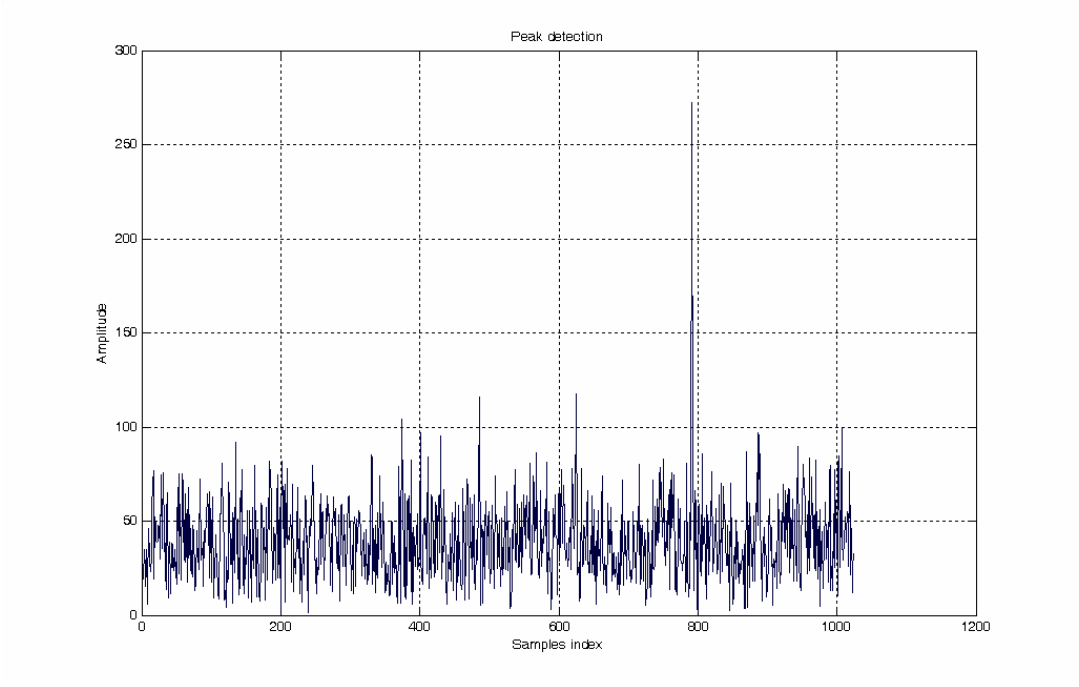


Figure 3.27 Symbol timing performance

From the Figure the symbol timing synchronization peak is at sample index 791 of the received data array. The plot has only included the received data samples before synchronization peak detection and samples from where a peak has detected. It was shown in chapter two that the first symbol in the transmission frame is the null symbol of sample length 664 and each OFDM symbol has the cyclic prefix sample of 126 (mode II) at the beginning of the symbol. The useful symbol duration does not include the cyclic prefix. Adding the null symbol samples and the cyclic prefix samples we obtain 790, thus the synchronisation peak is at exactly at starting point of the effective symbol duration of the phase reference symbol as expected.

b) Error analysis test

The error analysis test was carried out by analysing the bit error rate performance of the system in the presence of the AWGN channel, where a different noise level (variance) was used. For each noise level test an experiment was executed forty times in order to get enough statistical data. The analytical BER was calculated according to [28], which is also shown in the derivation

given in appendix section C. From [28] it was shown that π/4DQPSK performs 2.3dB worse than basic QPSK and BPSK. The BER and the SNR obtained are shown in Table 3.2.

SNR (dB)	Expected BER	Simulated BER
-7.6633	0.3270956	0.3962131
-5.9614	0.2929097	0.3564128
-2.8691	0.2183082	0.2648220
1.1343	0.1087137	0.1284722
2.3112	0.0789196	0.0888889
4.0324	0.0425430	0.046807
7.1122	0.0070499	0.0080838
11.0856	0.0000525	0.0000543

Table 3.2 Error analysis table

Both simulated and the expected results BER performance plots are shown in Figure 3.28.

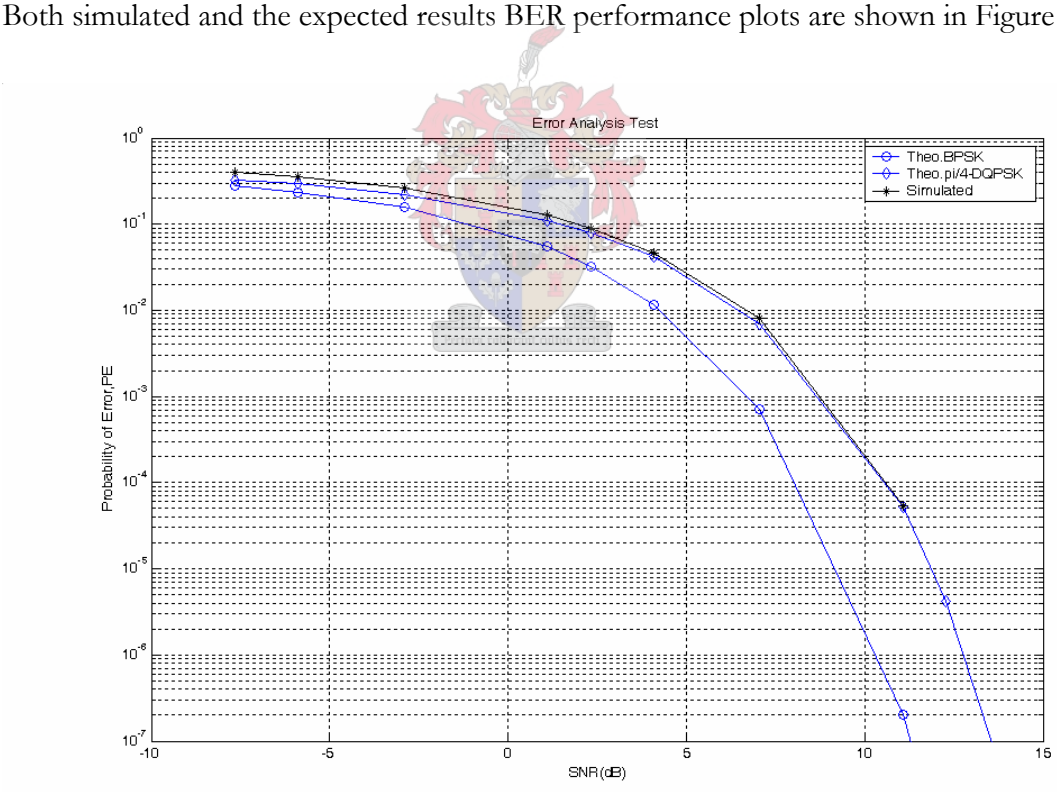


Figure 3.28 Error analysis plot

From the figure, we can see good agreement between the results of the simulated BER and the theoretical BER. The plots included the theoretical BER for BPSK because the BER performance for π/4-DQPSK was derived from the theoretical analysis for BPSK according to [28].

It is evident that the simulated and analytical BER are in good agreement. This proves that the simulated model is correctly implemented. Hence the simulated software worked correctly and real time implementation can be done accordingly. The next chapter will now describe the real time implementation in SDR.

Chapter 4

REAL TIME IMPLEMENTATION

4.1 Introduction

In this chapter, the SDR architecture developed at the university of Stellenbosch [27] is used to implement the DAB transmitter and receiver in real time. The DAB transmission mode II is implemented. Only the DAB system blocks used in the simulation for both transmitter and receiver are implemented in the SDR architecture. The chapter also covers an introduction to the SDR converters used to implement the transmitter and receiver, as well as the implementation problems and their solutions.

4.2 Introduction to SDR converters

The DAB system simulated in chapter 3 is made up of a number of functional blocks. To ensure the real time implementation in SDR a number of component blocks called converters [27] have to be constructed on the SDR architecture layer called the converter layer. This layer is responsible for all information signal processing functionality on the SDR system and is made up of a collection of converters.

A converter [30] is an atomic unit that performs a well-defined signal processing function. Thus, a converter receives data from source, processes data according to the defined algorithm and outputs a result. The converters have some input and output ports (see Figure 4.1). When a converter has finished processing its samples, it transfers the samples to the input port of the next converter.

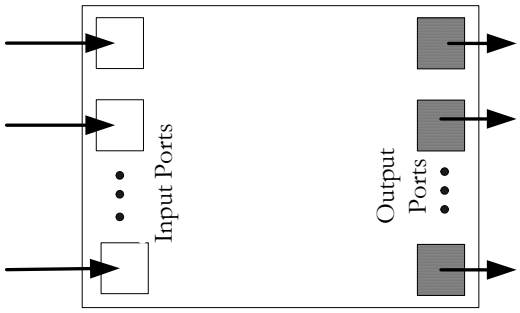


Figure 4.1 A basic converter representation

To write the processed data samples to the input port of the next converter, a write_output_port method should be used. Similarly to read data from the input port buffer, a read_input_port method should be used. To achieve reading and writing operations between converters, a well-defined interface method called " link " is used that connects an output to an input port. The details of these methods are clearly defined in [27], the following lines of code illustrates how the methods are used:

a) Writing sample to the next converter
   write_output_port(port_name, sample)
   port_name is the output block's port name
   sample is the sample that is passed on to the next block.

b) Reading data sample at the input port
   read_input_port(port_name)
   port_name is the input block's port name

c) Connecting the output port to the input port
   source module->link (output_port_number, destination_module, destination_portnumber)
   output_port_number is the port the samples are coming from
   destination_module is a RCPtr to the module to be linked to this one
   destination_portnumber is the port number of the module linked to this one.

4.3 Real time implementation considerations

In real time the processing in the transmitter and receiver needs to be carried out at great speed. Thus the converter must be processed as fast as possible. Remember both transmitter and receiver are implemented using a number of converters and each converter needs time to process its samples. To make the data processing as fast as possible in either transmitter or receiver a limited number of converters have to be constructed with each converter being implemented with a simple algorithm. Here the word "simple algorithm" means not a complex algorithm that takes a lot of time to execute but a simple algorithm that takes a little time to execute while yielding good results for the intended task. This can be achieved by combining some blocks' functionality used in the simulation to a single converter and implementing the functionality with as simple algorithms as possible. This will ensure that as little time as possible is spent on data transfer between converters, and moderate memory will be allocated to the system.

The operation of the system is based on transmitting a sequence of frames one at a time. Each frame has a structure as described in chapter 2. The data processing and manipulation depends on the order of the data samples in a frame. Thus we need to be sure of the content and the order of the data samples in a given transmission frame. When working with this structure it is better to create some vectors (buffers) with a transmission frame size within converters that will handle temporarily the input samples before being processed and resize them after their samples have been processed. This provides room and confidence for handling and manipulating the data samples of a given frame more efficiently and correctly, rather than depending on the converter input port buffer. Also by resizing the vectors to zero size after their samples have been processed, releases the memory that ensures proper memory usage.

We are generating and transmitting a complex base-band signal. The real and the imaginary part of the signal cannot be handled and transferred between converters using single port (i.e. one input port and one output port). The solution is to construct converters with two input ports and two output ports. One of the ports used to handle the transfer of the real part (in-phase) of the signal and the second used to handle the transfer of the imaginary part (quadrature) of the complex base-band signal.

4.4 Implementation overview

Both the transmitter and receiver were constructed using a number of converters that will be presented in the next sections. The converters making up both transmitter and receiver have two input ports and two output ports except the QPSK symbol mapper and QPSK symbol demapper. The QPSK symbol mapper in a transmitter has one input port to input serial data bits and two output ports. The QPSK symbol demapper in a receiver has two input ports and one output port to output serial data bits. The usage of two input ports and two output ports in the converters enable working with the complex signal in real time as described in the above sections and illustrated in Figure 4.2.

The block diagram shown in Figure 4.2 shows the real time implementation approach. The data generator in the figure is a converter that is responsible for generating binary data information and passing it to the transmitter input. At the transmitter, the converters that build up the transmitter modulate the input's binary data and produce two output streams namely in-phase (I) and quadrature (Q) (see Figure 4.2). The modulated data signal in both streams are passed to the data acquisition card (DAQ) comprises of an analogue to digital converter (ADC) and a Digital to analogue converter (DAC). The DAQ converts the inputs' digital samples from the transmitter to the analogue representation samples and then converts these analogue representations back to the digital samples. The digital samples from the DAQ are demodulated by the receiver's converters, which regenerate the original data.

Note: The DAQ was used for the real time testing purposes.

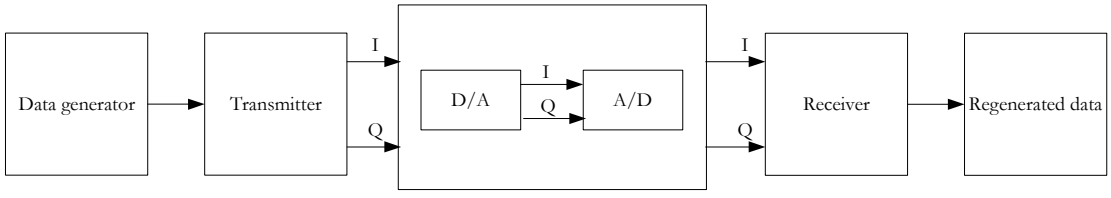


Figure 4.2 Real time implementation block diagram

4.5 DAB transmitter implementation in SDR

In order to implement the DAB transmitter in the SDR architecture a number of converters shown in Figure 4.3 were constructed. Each converter in the figure performs an independent function, that is reading samples from its input ports (port), processing them and transferring the processed samples to the input ports of the next converters (converter). The communications and links between converters are provided by the code given in section 4.2.

The algorithms implementing the transmitter's converters are similar to the algorithms used to implement a similar function block in the simulation, except that there are some additional features that will be discussed in the next subsections. The details of these algorithms have already been presented in chapter 3. In this section the added converters' functionalities, implementation problems and solutions, together with the converters operations, will be described.

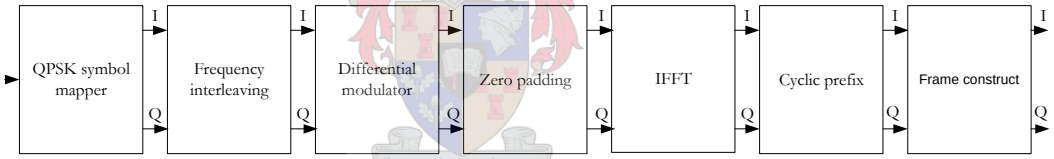


Figure 4.3 The converter used to implement the DAB transmitter in SDR architecture.

4.5.1 QPSK symbol mapper

As described in the above sections, this converter combines two functionalities. These functionalities are:

- Data partitioning into blocks
- QPSK symbol mapping

The converter reads data from its input port buffer provided there are enough data samples to constitute a transmission frame (remember we are working on frame basis). The samples

equivalent to a transmission frame size are read from the input port buffer one at a time, temporarily stored in the converter vector, processed and written to the input ports of the frequency interleaving. The processing performed is described next.

Data partitioning into blocks:

From a converter vector, blocks of 768-bit samples (mode II) are taken one at a time as described in chapter 3 section 3.4, based on first in, first out (FIFO) protocol. Each time a block of bits is taken, it is applied to the QPSK symbol mapping as described next.

QPSK symbol mapping:

QPSK maps bits into a complex symbol that can take one of the four possible values in the constellation plane. In real time we cannot work with a complex symbol represented by its real and imaginary part in one equation line. The solution is to treat the complex symbol by separating its real part and imaginary part and treat them separately both as the real component. The real part value corresponds to the in-phase component (I) of the complex symbol and the imaginary part corresponds to the quadrature (Q) component. These two components form the two output streams, namely in-phase (I) and quadrature l (Q) (see Figure 4.3).

From equation 3.3 in chapter 3, the in-phase (I_n) and quadrature (Q_n) components representing the complex symbol can be generated independently as follows:

$$I_n = \frac{1}{\sqrt{2}}(1-2.b_n) \quad for \quad n = 1,2,...,K \tag{4.1}$$

$$Q_{n-K} = \frac{1}{\sqrt{2}}(1-2.b_n) \quad for \; n = K+1,...,768 \tag{4.2}$$

where b_n represents a bit from a block of 768 bits samples that will constitute information for OFDM symbol and K is the total number of carriers used in the transmission (384 in mode II).

Here a block of 768 bits is mapped into K in-phase components and K quadrature components. The first half of the 768 bits in a block is used to generate K in-phase components and the last half used to generate K quadrature components. These form two

output streams of the converter described above. Note: the K symbol components from (4.1) and (4.2) form two separate QPSK symbol component blocks (I and Q).

After a bit in a block has been mapped, the resultant symbol component is written to the respective input port of the frequency interleaving. The in-phase components are transferred to the frequency interleaver through the in-phase output port and the quadrature components through the quadrature output port.

4.5.2 Frequency interleaving

After the symbol mapping has been carried out as explained in sub-section 4.5.1, the QPSK symbol component streams that are equivalent to transmission frame size become available at each input port of the frequency interleaver. All these symbol components are read from the two input ports and stored in two vectors, each having one transmission frame size. The first vector stores the in-phase components and the second vector stores the quadrature components. From each vector K (384) symbol components are taken one at a time and re-ordered according to the algorithm described in chapter 3. A similar interleaving is applied on each stream (I and Q) of components to ensure that the symbol components are interleaved equally. This assigns the symbol components to the respective sub-carriers. The result of the interleaving is written to the two respective input ports of the differential modulator.

4.5.3 Differential modulator

At this stage the information in the system belongs to the QPSK constellation that is, information is carried in absolute phase. But the information is supposed to be carried in phase difference [5]. It is the task of this converter to differentially modulate the inputs' QPSK sub-carriers that ensure that the information is carried in phase difference as presented in chapter 3.

The converter is fed with two input streams (I and Q) of the interleaved QPSK symbol components. The symbol components from the two input streams are only read provided there are enough samples to constitute a transmission frame. One transmission frame sample size is

read at a time from each input port and temporarily stored in two respective vectors before differential modulation is applied.

The converter combines two functionalities that are performing the differential modulation and generating the phase reference symbol. Recall, as described in chapter 3, the differential modulation requires a phase reference. The phase reference symbol is generated as described in chapter 3, to serve this purpose. The generated phase reference symbol components are written to the two input ports of the zero padding converter and their copies (I and Q) are stored in two vectors for later use in the differential modulation. Remember the phase reference symbol appears first in the transmission frame that is why it is generated and written at the output ports before other information samples are written. Now the number of symbol blocks generated that form OFDM symbols, becomes 76 in total. Thus the output of this converter is a complete transmission frame without a null symbol.

The generation of the differential symbol is according to the π/4-DQPSK signal mapping presented in equation 3.8, chapter 3. The in-phase differential components (Iz) and quadrature differential components (Qz) are generated as illustrated in equation 4.3. We can see from equation 4.3 how the two differential symbol components are handled separately.

$$\begin{aligned} Iz_{l,k} &= I_{l,k} Iz_{l-1,k} - Q_{l,k} Qz_{l-1,k} \\ Qz_{l,k} &= Q_{l,k} Iz_{l-1,k} + I_{l,k} Qz_{l-1,k} \\ &for\ l = 2,3,...,L \\ &and\ k = 1,2,...,K \end{aligned} \tag{4.3}$$

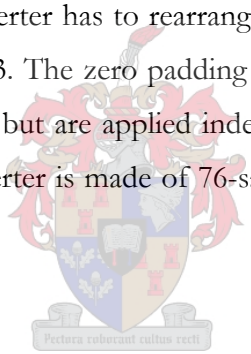
where I is the QPSK symbol block from the input samples with K in-phase components, Q is the QPSK symbol block from the input samples with K quadrature components, l is the block index of K symbol components each of which represents samples for an OFDM symbol (L =76), and K is the total number of carriers.

The phase reference symbol is used to provide the initial phase reference (i.e. provides the initial Iz_{l-1} and Qz_{l-1}). From the two vectors storing the inputs' QPSK symbol components (I and Q), K symbol components from each vector are taken one at a time and applied to the differential modulation as illustrated in equation 4.3. This results in two differential QPSK symbol component blocks (Iz and Qz) each with K components. The result of the modulation (Iz

and Qz) for each QPSK symbol block is written to the two respective input ports of the zero padding and a copy of their components is stored in phase reference vectors (Iz_{l-1} and Qz_{l-1}) for usage in the differential modulation of the next K QPSK symbol components. All the vectors are resized to zero after the modulation ready for receiving the next frame components.

4.5.4 Zero padding

The two input streams from the differential modulator are made of 76-differential QPSK symbol blocks each with K (384, mode II) components where the first K components in each stream belong to the phase reference symbol. As described in chapter 2, the IFFT/FFT algorithms work more efficiently if the number of input samples is a power of 2. It is the task of this converter to add zeros to each differential QPSK symbol block so that it fits 512-point IFFT (mode II). Also the converter has to rearrange the carrier samples in a manner similar to what was discussed in chapter 3. The zero padding and rearrangements are applied in a similar way to both streams (I and Q) but are applied independently to each stream. Each of the two outputs (I and Q) of this converter is made of 76-sample blocks each with 512 samples, which feeds the IFFT converter.



4.5.5 IFFT

This is the main converter of the transmitter. The equation for generating the OFDM symbols described in chapter 2 is implemented in this converter. In short the converter generates the real time DAB main signal. The orthogonal carriers for an OFDM symbol are automatically generated here. The converter was implemented with a suitable IFFT algorithm from [24] that uses the two inputs (I and Q) to generate the two output streams that form the real and imaginary part of the DAB complex base-band signal.

The converter reads the zero-padded samples from each of the two input ports (I and Q) when there are enough samples equivalent to a transmission frame size (76 x 512). These samples are read and stored in two vectors each with a transmission frame size. One of the vectors stores I-phase components and the second vector stores quadrature components. From each vector 512

sample components are taken one at a time and simultaneously applied to the IFFT algorithm that generates an OFDM symbol from these samples. This result in two outputs (I and Q), one forms the real part of the base-band DAB signal and the second forms the imaginary part of the base-band DAB signal. The results are written to the respective two input ports of the cyclic prefix converter.

4.5.6 Cyclic prefix

This is the last converter of the transmitter. Each of the two input streams (I and Q) from the IFFT converter is made of 76-sample blocks each with 512 samples. The converter reads samples, equivalent to one transmission frame size, one at a time from each input port and stores them in two respective vectors. From each vector 512 samples forming an OFDM symbol are taken separately and guard samples are added as described in chapter 3. Remember, until this stage the transmission frame is correctly packed and with required structure but yet not ready for transmission. The transmission frame is missing the first symbol that is a null symbol. The converter generates and writes a number of zeros equivalent to the null symbol period to the output ports before other samples in the received transmission frame. The zeros generated serve as the null symbol. Writing these zeros to the outputs before other samples ensures that the null symbol takes a first position in the transmission frame. An equal amount of zeros representing the null symbol samples are written to the two output ports.

4.5.7 Frame construct

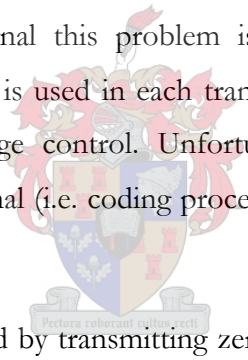
The work of this thesis was developed in base-band signal. As presented in chapter 3, the DAC/ADC and the RF section were not studied. But for the system testing purpose in real time the DAQ card comprising of the DAC and ADC was used. When using this two problems were encountered:

- The generated signal has higher dynamic range and some of the signal peak amplitudes are greater than the input range for the DAQ (at DAC inputs) used. This problem has to be solved because it results in signal clipping, hence signal distortions.

- The DAQ card used has internal buffers that need to be emptied at the end of the transmission frame. Otherwise, it retains some information data in its buffers that leads to the output signal being different from the input signal.

This is the hardware problem that was not studied. To solve the first problem for the testing purposes, the transmitted signal amplitude was scaled down by a certain factor that was obtained practically. It was assumed that the receiver knows this scaling factor, thus at the receiver the received signal was expanded by the same factor. But this did not solve all the problems of the input range at the DAC. This is because scaling down the signal has a limit in order to get a good output signal, so it is the matter of compromise between scaling down the signal and the DAC input range. The solution is to scale down the signal amplitude peaks to a certain limit obtained practically and that yields good results and to clip some of the signal amplitude peaks (set them to the maximum DAC input range) that exceed the DAC input range.

In generating a real DAB signal this problem is handled by the use of the Programme Associated Data (PAD), which is used in each transmitted audio frame. One of the functions for the PAD is dynamic range control. Unfortunately all the processes involved in the generation of the real DAB signal (i.e. coding process) were not studied in this thesis. Thus the above solution was used.



The second problem was solved by transmitting zeros equal to the size of the DAQ buffers at the end of the transmission frame. This ensures that all the transmission frame data have been sent out the DAQ buffers and the buffers are full of zeros.

To solve these two problems the frame construct converter was constructed. The implemented converter algorithm takes care of these two problems and writes the sample signal from the cyclic prefix to the two inputs (I and Q) of the DAQ.

4.6 DAB receiver implementation in SDR

The implementation of the DAB receiver in real time was accomplished by constructing a number of converters in the SDR architecture described in section 4.2. Figure 4.4 illustrates the converter implementing the receiver. The complex base-band signal is received at the receiver from the DAQ in two paths. One path for the real part and the second path for the imaginary part.

The converters details have already presented in the above section 4.2. The algorithms implementing the receiver converters are basically an inverse of the algorithms implementing the transmitter converter. The algorithm's details have already been explained in chapter 3, in this section additional features and some changes in the algorithms are discussed.

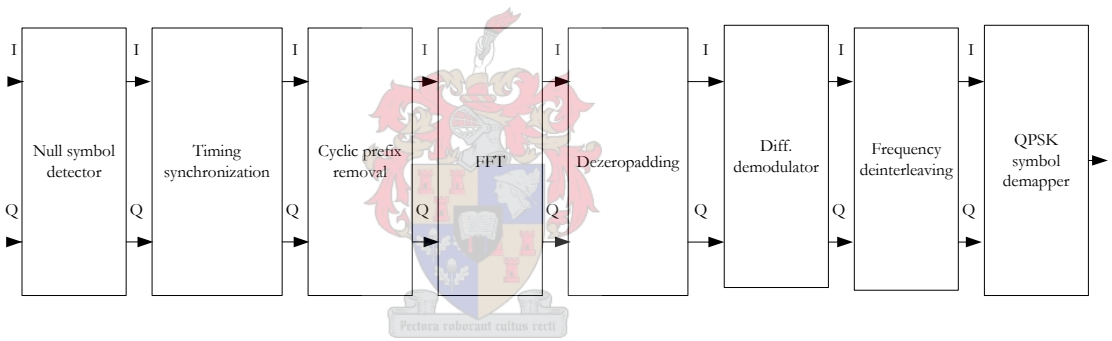


Figure 4.4 The converter used to implement the DAB receiver in SDR.

4.6.1 Null symbol detector

The null symbol detector is the first converter of the transmitter. It estimates the start of the transmission frame in the received data signal. The estimate is achieved by measuring the average power for the null symbol in the incoming data signal. The data samples equivalent to the null symbol sample size are used in the average power measurement. The measurement is done as illustrated in the following expression:

$$avg.power = \frac{1}{N}\sum_{n=1}^{N}\left(X_i(n)^2 + X_q(n)^2\right) \tag{4.4}$$

where $avg.power$ is the measured average power, X_i and X_q are the real and imaginary part of the received data signal and N is the total number of samples used in the measurement that is equivalent to the null symbol sample size (N=664 mode II).

In the implementation, this converter keeps listening to the incoming data signal. When there are incoming signal samples at the input ports, the converter passes these samples to the input port of the timing synchronization converter and uses a copy of these samples to compute the average signal power as illustrated in equation 4.4. The measured average power is always compared to the threshold level in order to determine the start of the frame. The threshold level used in this implementation is half of the transmitted signal power. If the measured power is less than the threshold level, a null symbol is detected, hence the start of the frame.

It is very interesting to understand why the converter estimates the start of the frame and passes the entire received data sample to the next converter. The answer is that the estimates do not work well in a low SNR environment. This is because when an amount of noise added to the signal (null symbol part) lead to estimated average noise power that is greater than the threshold level the detection fails. We need our synchronization to be SNR independent. The solution to this is to get a rough estimate of the start of the frame using the null symbol detector but not relying on this estimate, instead we use the symbol timing synchronization to provide accurate frame timing and symbol timing synchronization simultaneously. Thus the null symbol's detector is used to provide rough estimates of the start of the frame and passes the entire received data signal samples to the timing synchronization converter for accurate timing estimation as presented in the next subsection.

4.6.2 Timing synchronization

The timing synchronization converter is responsible for symbol timing synchronization and frame timing. The timing information is obtained from the accurate measurement of the start of the phase reference symbol in the incoming data signal. The implemented timing synchronization loop follows the following steps:

1. Read 512-samples of the incoming data from each of the two input ports (I and Q) and store them in two respective vectors.
2. Use the data samples from the two vectors in 1 to determine the channel impulse response (CIR).
3. Search for the desired synchronization peak from the result obtained in 2.
4. Determine the start of the phase reference symbol if the desired synchronization peak is found. Otherwise, go back to 1.
5. If the timing synchronization is established, read the desired data samples from the two input ports (I and Q) and pass them to the cyclic prefix removal through the two respective output ports (I and Q), including samples of the useful part of the phase reference symbol.
6. If the received data samples equivalent to one transmission frame size have been transferred to the cyclic prefix removal after synchronization, go back to 1.

The timing synchronization loop in this implementation always keeps on searching if there are enough samples at the two input ports for determining the channel impulse response (CIR). The 512-samples are used because the implementation uses 512-point FFT. When there are enough samples the CIR is determined as described in chapter 3 and impulse signal is compared to the set threshold level. This determines the start of the phase reference symbol. All the samples read from the received data before the phase reference symbol found are discarded.

As described in chapter 3, the synchronization peak is always at the start of the useful part of the phase reference symbol period. When synchronization is achieved the useful parts samples for phase reference symbol are written to the cyclic prefix removal followed by the desired received data information samples. The phase reference symbol samples are passed to the next converter because they carry the initial phase reference information required for the differential demodulation.

During transfer of the desired received data samples to the cyclic prefix removal, the synchronization loop is switched off and a counter is set on to count the desired samples (one transmission frame samples excluding the null symbol samples and guard interval samples of the phase reference symbol). As soon as the desired samples have been transferred the loop is again switched on. This is done to ensure that the data transfer is done as fast as possible since the

synchronization calculations consume some time. The amounts of the desired samples in a transmission frame after synchronization are known according to the frame structure. Thus the setting of the counter becomes practicable.

4.6.3 Cyclic prefix removal

The timing synchronization converter passes only the received data signal samples equivalent to one-transmission frame size excluding the null symbol samples and the guard interval samples for the phase reference symbol. These become available at the two input ports of this converter after timing synchronization has been achieved. The converter removes the guard interval samples added at the transmitter.

The procedure implementing this converter is as follows:

1. Read samples from each input port that are equivalent to one transmission frame size at a time as described in the above paragraph and store them in two respective vectors each with the size of one transmission frame size excluding the null symbol samples and phase reference symbol guard interval samples.
2. Write the first 512-sample from each vector to the two respective output ports (these samples belong to the phase reference symbol from which its guard samples have been removed in the previous converter).
3. Take a sample block of 638-samples from each vector and write the last 512-samples to the corresponding output port since the first 126 samples of this block belong to the guard interval samples.
4. If all the sample blocks from each vector have been used, go back to 1. Otherwise, go back to 3.

4.6.4 FFT

After cyclic prefix removal, 76-sample blocks each with 512 samples will be available at each input port of this converter. The converter reads these samples from the two input ports and stores them temporarily in the two vector one for the real part and the second for the imaginary

part of the received data signal. From each vector 512 samples are taken one at a time and applied simultaneously to the implemented FFT algorithm. The results of the FFT computation are written to the zero padding removal. This converts back the samples in each sample block to the frequency domain representation.

4.6.5 Zero padding removal

The real and the imaginary parts of the samples from the FFT converter are fed to the two respective input ports of this converter. Each input sample stream to the converter is made of 76-sample blocks each with 512 samples. The converter reads all of these samples and stores them temporarily in two vector one for the real samples and the second for the imaginary samples. From each vector 512 samples are taken one at a time (remember how zero padding was carried out), the added zeros at the transmitter are removed and the carrier's samples are rearranged as described in chapter 3. The output results are the carrier's samples as presented at the input of the zero padding and these are written to the next converter through the two respective output ports (I and Q).

4.6.6 Differential demodulator

The input samples at each of the two input ports of this converter are made of 76-sample blocks each with 384-differential carrier samples. The converter reads these input samples from each input port and temporarily stores them separately in two vectors each with one transmission frame size (76 x 384). From each vector 384-samples are taken one at a time and applied to the differential demodulation according to equation 3.16, chapter 3, and illustrated in equation 4.5. The first 384-samples in each input stream belongs to the phase reference symbol that provides the initial phase reference (Iz_{l-1} and Qz_{l-1}). The rest of the phase reference is provided by the previous sample block for the current sample block differential demodulation (see equation 4.5). The output results are in-phase and quadrature components of the QPSK symbols that are written to the next converter through the two separate output ports.

$$
\begin{aligned}
&I_{l,k} = Iz_{l,k}Iz_{l-1,k} + Qz_lQz_{l-1,k} \\
&Q_{l,k} = Qz_{l,k}Iz_{l-1,k} - Iz_{l,k}Qz_{l-1,k} \\
&for\ l = 2,3,...,L \\
&and\ k = 1,2,...,K
\end{aligned}
\tag{4.5}
$$

where I is the QPSK symbol block with K (384) in-phase components, Q is the QPSK symbol block with K quadrature components, Iz is the differential QPSK symbol block with K real samples of the differential QPSK symbols, Qz is the differential QPSK symbol block with K imaginary samples of the differential QPSK symbols, K is the total number of carriers (384 mode II) used in the transmission and L is the total number of sample blocks (76 mode II).

4.6.7 Frequency deinterleaving

Each of the two input streams to this converter is made of 75-QPSK symbol blocks each with 384 QPSK symbols. This convert reverses the process of the frequency interleaving at the transmitter. It rearranges the QPSK symbol components in each QPSK symbol block to the original order as before frequency interleaving, as described in chapter 3. The algorithm implementing this converter is an inverse of the algorithm implementing the frequency interleaving.

4.6.8 QPSK symbol demapper

This is the last converter of the receiver. The output of this converter should be the original data as at the input of the transmitter. The converter receives the QPSK symbol streams at its two respective input ports. One of the inputs receives the in-phase components and the second input receives the quadrature components. At each input port there are 75-QPSK symbol blocks each with 384-QPSK symbol components at a time. The de-mapping steps implementing the converter are as follows:

1. Read the in-phase and quadrature components from the two input ports when the available components at each port are equivalent to one transmission frame size and

store them in two vectors one for in-phase components and the second for quadrature components.

2. From each vector take a block of 384-sample components.
3. De-map the in-phase component samples from one of the blocks obtained in 2 and write the decoded bit to the output port
4. De-map the quadrature component samples from one of the blocks obtained in 2 and write the decoded bit to the output port used in 3.
5. If all the sample components of the vectors in 1 have been decoded go back to 1. Otherwise, go back to 2.

The sample components of the blocks in 2 above have the corresponding real and imaginary parts of the complex QPSK symbols. Remember from section 4.5.1, these two-sample blocks decode in a block of 768 bits, the first half of the block being contributed by the decoded in-phase component and the second half contributed by the decode quadrature components. That is why the in-phase components are decoded first and written to output port, followed by the quadrature components. This ensures that the decoded bits will be in the same order as the input of the transmitter.

According to the mapping used in section 4.5.1, when a received symbol component (either in-phase or quadrature) has a negative value the component is decoded as bit '1' and when a symbol components has a positive value the components is decoded as bit '0'. In this implementation the decoded bits were stored in a vector and their copies were written to the next converter block for error checking.

4.7 Conclusion

The implementation for both transmitter and receiver in SDR is described in this chapter. The transmitter generated the complex base-band DAB signal at its two output ports, one of the ports for the real and the second port for the imaginary part of the complex base-band signal. These were fed into two corresponding inputs of the DAQ. The two outputs from the DAQ provided the inputs to the receiver. The receiver demodulated the signal. The output of the receiver was equivalent to the input at the transmitter. The symbol timing synchronization

worked very well. The real time test results showed a negligible implementation loss of about 0.3dB. Thus implementation was successful. The results are described in chapter 5. The serious problem encountered was the distortion introduced due to the signal clipping in DAQ. This was solved as described in this chapter, but another solution can be the use of the DAC/ADC with large dynamic range, although if the whole signal generation process is considered, the dynamic range controls method and forward error correction used in the system will give promising results.

Chapter 5

IMPLEMENTATION EVALUATION AND RESULTS

5.1 Introduction

This chapter shows tests performed to evaluate the performance of the implemented system and the results obtained in both simulation and real time implementation. The results for each test are discussed.

5.2 Simulated symbol timing synchronization performance

In this section the symbol timing synchronization performance of the implemented system is evaluated. The relationship between symbol timing synchronization and the receiver tuning is investigated and discussed.

The objective of the experiment is to investigate the performance of the implemented timing synchronization on the incoming signal at the receiver. This verifies the performance of the timing synchronization at any stage of the incoming signal. In real life the receiver synchronization does not follow whether the receiver is tuned when the incoming signal is either at the start of the transmission frame or at any part of the frame. The receiver synchronization should look for the synchronization information in the incoming signal and establish synchronization. Hence the symbol timing synchronization should synchronize perfectly in either case.

5.2.1 Experimental setup

The simulated system model presented in Figure 3.2 was used for symbol timing synchronization performance test. The random binary numbers constituting three OFDM symbols were generated. Two transmission frames were used in the test, each made of three

OFDM symbols having similar binary numbers generated. The simulated DAB signal was generated as described in chapter 3. The symbol timing synchronization at the receiver was acquired according to the description provided in chapter 3.

5.2.2 Results of the experiment

The real part of the simulated DAB base-band signal and the symbol timing synchronization plots are shown in Figure 5.1. In the figure the first transmission frame was transmitted without including the synchronization part (null symbol and the phase reference symbol). The second transmission frame included the synchronization part. The synchronization part of the first transmission frame was chopped off to reflect what can happen in the real world when a receiver is tuned. The receiver synchronization should be independent of at what stage of the incoming signal a receiver is tuned but it should look for the synchronization information on the incoming signal. When the synchronization information is found, the synchronization should be established.

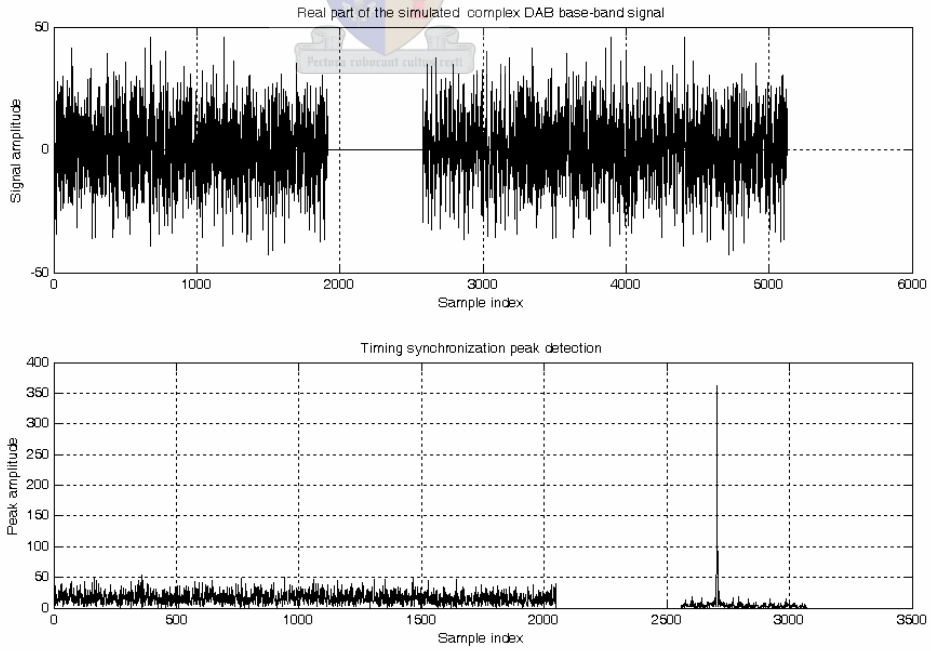
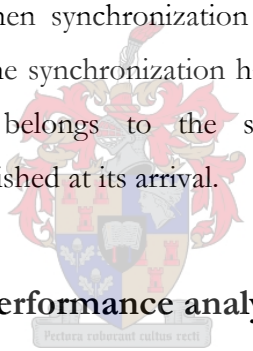


Figure 5.1 The symbol timing synchronization performance in real world

From the figure we see the desired symbol timing synchronization peak. The peak is exactly at the starting point of the useful symbol period for the phase reference of the second transmission frame. This can be verified by considering the structure of the two transmission frames, which have arrived at the receiver. The first transmission frame is made of three OFDM symbols each consisting of 638 samples (useful symbol samples (512) plus guard interval samples (126)). Between the two frames before the start of the useful part of the phase reference symbol of the second transmission frame, we expect the null symbol samples (664) and the guard interval samples (126) added to phase reference symbol. From the figure it can be seen that the symbol timing synchronization peak is at 2705 sample index, which is the starting point of the useful symbol part for the phase reference symbol from the transmitted data, as expected.

This is the real world situation, when a receiver is tuned. The receiver is expected to start decoding the received data when synchronization has been established. The incoming data arrived at the receiver before the synchronization has established are not decoded. In this case the decoded received data belongs to the second transmission frame because the synchronization has been established at its arrival.



5.3 Bit Error Rate performance analysis

The performance analysis test of the implemented DAB system in SDR is conducted in real time. The relationships between the theoretical BER and the practical BER are investigated and discussed.

The objective of this experiment is to investigate the performance of the implemented system in real time in the presence of AWGN.

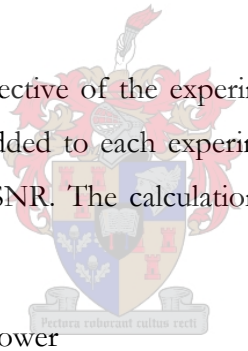
5.3.1 Experimental setup

The real time implemented system shown in Figure 4.2 is used to perform the performance test. Random binary numbers equivalent to a transmission frame size are generated and modulate in the transmitter. The output base-band signal (I and Q) from the transmitter is converted into

analogue domain using a DAQ card. The analogue base-band signal (I and Q) from the DAQ card is added to the noise. The noisy base-band signal (I and Q) is converted back to the digital domain using the same DAQ card. The digital base-band signal from the DAQ card is demodulated in the receiver. The process is repeated, and each time the noise voltage added to the signal is increased. Both the transmitter and receiver program are implemented in one computer (PC) with 1600MHz CPU, 256kB cache memory and 256 MB RAM.

The digital-to-analogue conversion and analogue-to-digital conversion is done using the DAQ card as explained in the above paragraph. The DAQ card used comprises both DAC and ADC. The base-band signal is sampled in the DAQ at 4096Hz for both I and Q signal. Two random noise generators are used to add noise to the signal from the two output streams (I and Q) of the DAC. The noise is added to the signal using a summer made of non-inverting OP AMP. A True RMS voltmeter is used to ensure that an equal amount of noise voltage is added to the two channels.

In order to accomplish the objective of the experiment, the transmitted signal average power and the noise average power added to each experiment have to be determined. These will be used in the calculation of the SNR. The calculation of both average powers are as illustrated below:



1) Transmitted signal average power

We calculate the transmitted signal average power from [23]:

$$Avg.signal\ power = \frac{1}{N}\sum_{n=1}^{N} x^2(n) \qquad for\ \ n = 1,2,...,N \tag{5.1}$$

where N is the total number of samples belonging to the useful symbol period.

The transmitted base-band signal is made of two streams named in-phase (I) and quadrature (Q). The average signal power is given by:

$$Avg.signal\ power = \frac{1}{N}\sum_{n=1}^{N}\left(x_i^2(n) + x_q^2(n)\right) \quad for\ n = 1,2,...,N \tag{5.2}$$

where subscripts i and q indicates the in-phase and quadrature sample components respectively.

From (5.2) we obtained the transmitted signal average power of 384 watts.

2) Average noise power measurement

The average noise power is calculated using equation 5.2 where 664 samples are used. The measurement is done in the null symbol part of the received frame. There is no data transmitted in this part, zero average power is expected at the receiver when there is no noise added to the signal. Each time the noise voltages are varied, the average noise power is calculated.

5.3.2 Results of the experiment

Table 5.1 shows the results obtained for the BER for both real time and simulation tests. The noise levels used in the simulation are equivalent to the noise voltages used in the real time test.

Noise Power (Watts)	SNR (dB)	Theoretical Bit Error Rate (BER)	Real Time Bit Error Rate (BER)	Simulated Bit Error Rate (BER)
0.969	24.217	$1.37\text{x}10^{-69}$	0	0
4.841	17.234	$1.76\text{x}10^{-15}$	0	0
8.582	14.748	$1.69\text{x}10^{-9}$	0	0
20.314	11.004	$6.09\text{x}10^{-5}$	$6.91\text{x}10^{-5}$	$6.51\text{x}10^{-5}$
30.790	9.198	$90.10\text{x}10^{-5}$	$115.64\text{x}10^{-5}$	$108.51\text{x}10^{-5}$
41.685	7.883	$365.52\text{x}10^{-5}$	$446.2\text{x}10^{-5}$	$434.03\text{x}10^{-5}$
53.999	6.759	$921.73\text{x}10^{-5}$	$1081.60\text{x}10^{-5}$	$983.07\text{x}10^{-5}$
67.435	5.794	$1747.50\text{x}10^{-5}$	$2140.63\text{x}10^{-5}$	$1924.91\text{x}10^{-5}$
79.599	5.073	$2612.12\text{x}10^{-5}$	$3237.85\text{x}10^{-5}$	$2957.90\text{x}10^{-5}$
98.525	4.147	$4051.34\text{x}10^{-5}$	$5239.58\text{x}10^{-5}$	$4513.89\text{x}10^{-5}$
122.866	3.188	$5909.60\text{x}10^{-5}$	$6677.08\text{x}10^{-5}$	$6903.21\text{x}10^{-5}$
139.956	2.622	$7160.91\text{x}10^{-5}$	$8723.96\text{x}10^{-5}$	$8355.03\text{x}10^{-5}$
157.042	2.102	$8399.31\text{x}10^{-5}$	$10028.12\text{x}10^{-5}$	$9461.81\text{x}10^{-5}$
191.042	1.271	$10511.03\text{x}10^{-5}$	$12293.10\text{x}10^{-5}$	$12261.28\text{x}10^{-5}$
308.600	-0.812	$16210.01\text{x}10^{-5}$	$21100.00\text{x}10^{-5}$	$19385.85\text{x}10^{-5}$

Table 5.1 Performance error analysis table.

Figures 5.2 and 5.3 show both the expected and practical bit error rate (BER) curves of the implemented DAB system. In both cases the theoretical BER is calculated according to [28], which is also shown in the derivation given in appendix section C. From [28] it was shown that π/4DQPSK performs 2.3dB worse than basic QPSK and BPSK.

In Figure 5.2 both simulated and real time BER are shown. Figure 5.2 includes the theoretical BER for BPSK showing its relation to the modulation scheme (π/4DQPSK) used in the implementation according to [28]. The simulated and real time curves lie almost on top of each other. In both figures the implementation loss is roughly 0.3dB. It is evident from these figures that the theoretical and the practical BER are in good agreement. Hence the real time software worked correctly.

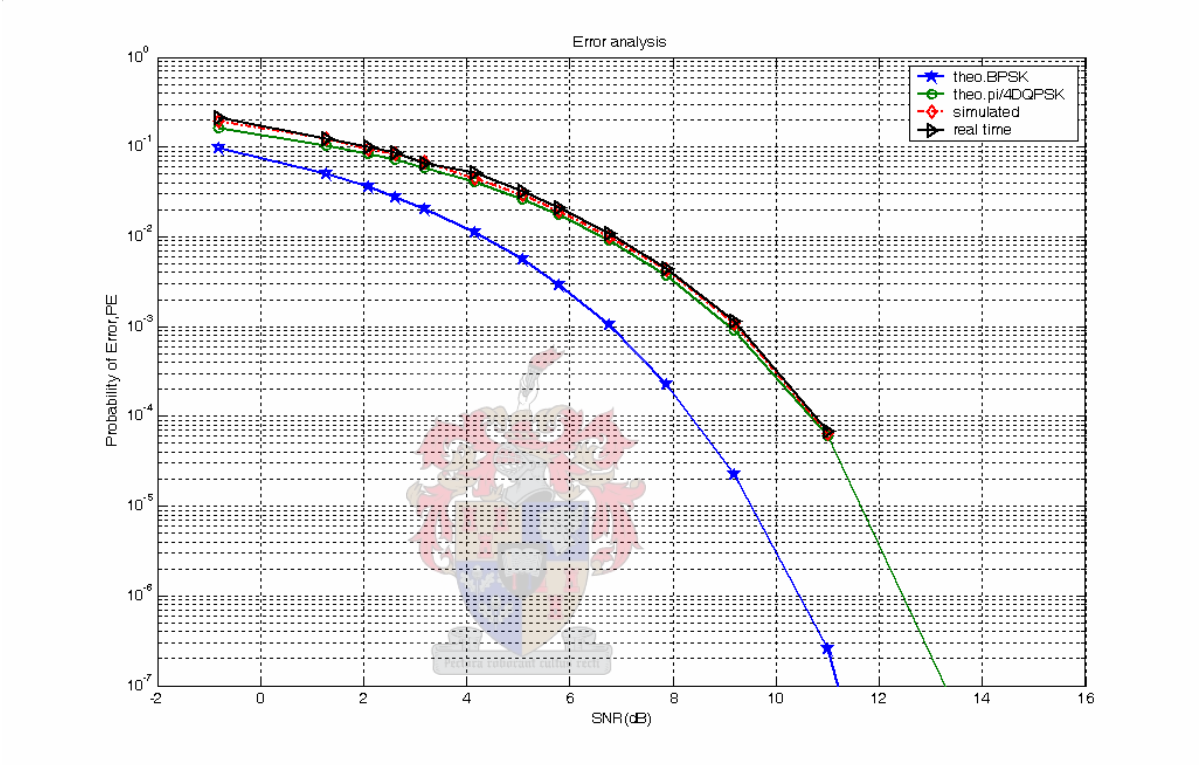


Figure 5.2 Performance error analysis simulated Vs Real time results.

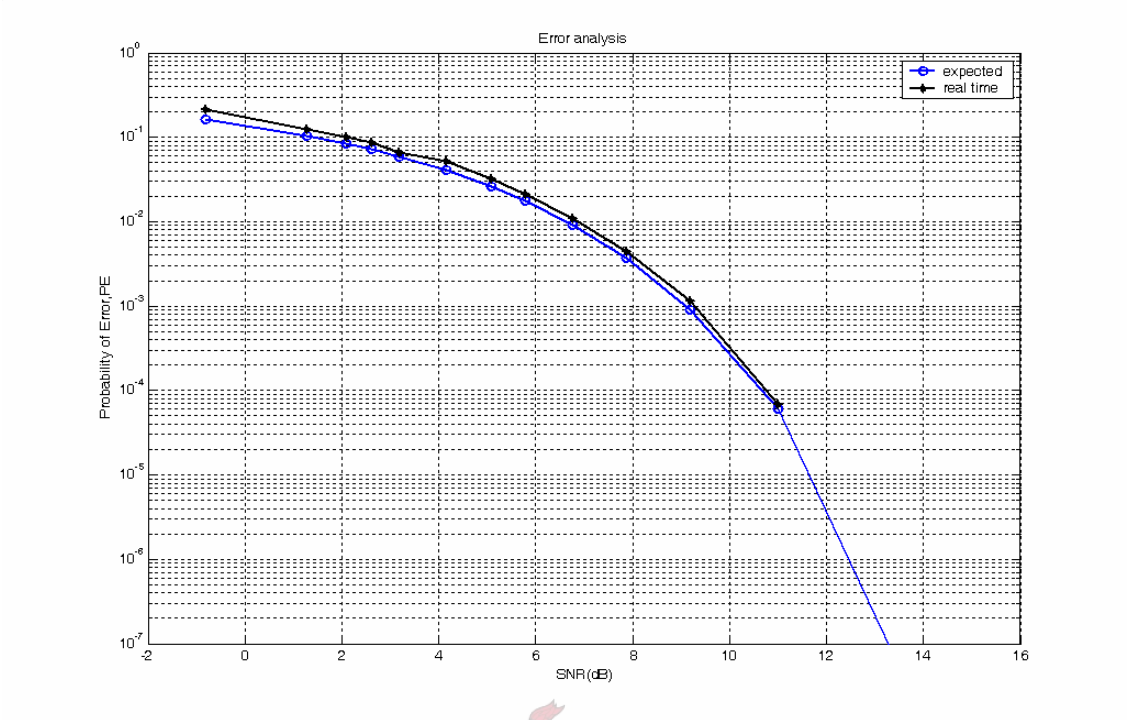


Figure 5.3 Real time performance error analysis plot.

5.4 Transmission time and processing time measurements

In this section the transmission frame time and the processing time are measured and the results are discussed.

The objective is to investigate the processing power required in real time and to compare the practical transmission frame time with the analytical estimation.

5.4.1 Experimental setup

The real time system implemented shown in Figure 4.2 was used to perform this test in real time. Binary numbers constituting one transmission frame were generated. A complete transmission frame was modulated at the transmitter. Both the digital-to-analogue and analogue-to-digital conversions were done using the DAQ card as explained in the above sections at the sampling rate of 4096Hz. The base-band signal (I and Q) is demodulated at the receiver. A

digital oscilloscope was used to measure the transmission frame time the signal spent on the channel. The Linux time command function was used determine the processing speed (time). Both the transmitter and receiver program were implemented in one PC with 1600MHz CPU speed, 256kB cache memory and 256MB RAM.

5.4.2 Results

Table 5.2 shows the expected transmission frame time according to the specification used in the implementation test. Table 5.3 shows the practical measurement results of the frame transmission time and the processing time used.

In the results, the analytical and practical transmission frame time measured shows a good agreement. The only small difference is because of the practical measurement error. The processing time measured shows how much time was taken to generate, modulate, transmit and decode a transmission frame. This is over all processing time of the code for both transmitter and receiver, plus the transmission frame time.

Expected analytical results

Particular	Formulae	
Total transmitted samples (excluding null symbol)	638 x 76	48488
Transmission frame time (T_F)	$\frac{total\ transmitted\ samples}{sampling\ frequency}$	11.84 sec
Total symbol duration (T_S)	$\frac{T_F}{No.symbol\ /\ frame}$	155.79ms
Useful symbol duration (T_u)		125.02ms
Carrier separation	$\frac{1}{T_u}$	8Hz
Transmitted signal bandwidth	384 x 8Hz	3.072kHz

Table 5.2 Expected analytical transmission time.

The processing time measured provides a rough indication of the processing power required when we are thinking of increasing the processing speed. For example in these tests the over-all processing time is about 16.07 seconds with 1600MHz CPU. In real implementation the sampling rate of 4096Hz cannot be used. The sampling rate used is 2.048MHz [5][25]. It can be seen from these two sampling rates that the data needs to be sampled 500 times faster than the present sampling rate used. We cannot use PCs for the real implementation because PCs are not fast enough to handle the real implementation. We need a faster and dedicated processor such as a DSP processor to achieve the real implementation. The fact that both transmitter and receiver algorithms run together at 500 times slower than the real time on a 1600MHz PC, gives an indication of the execution speed that provides a base of the processing power required when thinking of increasing the processing speed.

Practical measurements:

Measurement	Value
Transmission frame time on the channel	11.9 seconds
Total time used to generate, transmit and recover a transmission frame	16.07 seconds

Table 5.3 Practical transmission time and processing speed measured.

5.5 Conclusion

In this chapter the performance of the implemented system was evaluated. The test results for both real time and simulation were discussed in this chapter. The timing synchronization of the system worked well and enabled the correct recovery of the desired data at the receiver. The bit error rate of the system proved that the implemented system worked well in real time. A rough indication of the processing power required to provide a base for what processing power is required when thinking of increasing the processing speed, was discussed.

Chapter 6

CONCLUSION

This chapter gives the conclusion of the work done and offers some suggestions for future work that can be carried out.

6.1 Concluding remarks

In the DAB system, a good BER for audio is considered to be 10^{-4} [25]. From the results obtained during tests, a SNR of about 10.77dB provided the considered BER. This gives an indication of how much data error control protection should be included in the channel coding. At a SNR less than 10.77dB, the error control protection is crucial for the better performance of the system. At a SNR greater than 10.77dB and considering the test environment used, the implemented system is expected to provide good audio quality even when no error protection control is used.

6.2 Final conclusion

The DAB system was successfully implemented in SDR. The symbol timing synchronization worked well. A 0.3dB implementation loss was obtained. These facts prove the thesis goal was achieved. The perfect DAB modulator and demodulator for transmission mode II were added to the SDR library.

6.3 Future work

Future work can be done in the following areas:

1. In the system performance analysis, AWGN channel was used. The analysis can be expanded to Rayleigh fading channel, where the system performance can be analysed with the mobile receivers.

2. In all the implementation work, a perfect working hardware was assumed. A further study on the hardware should be done in order to investigate their effect on the DAB signal. For example a study on the automatic frequency control (AFC), RF section and ADC will facilitate the implementation for the carrier synchronization in the synchronization block.
3. A complete DAB signal generation should be implemented. The system should include the channel coding part and be able to generate a real DAB signal and demodulate.
4. The DAB transmission mode II was implemented. In order to be able to work with all DAB frequency ranges other transmission modes should be implemented.

Bibliography

[1] F. Kozamernik, "Digital Audio Broadcasting – radio now and for the future," ***EBU Technical Review no. 265***, Autumn 1995.

[2] T. O'Leary, "Terrestrial Digital Audio Broadcasting in Europe," ***EBU Technical Review***, Spring 1993.

[3] Stephen Baily, "A Technical Overview of Digital Radio," ***BBC Research and Development***, Kingswood Warren, Tadworth, KT20 6NP, United Kingdom.

[4] Pohlmann Ken, "Principles of Digital Audio," McGraw-Hill, New York, chapter 2.3,2000.

[5] ETS 300 401,"Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers," ETSI, second edition, May 1997.

[6] "EUREKA – 147 – Digital Audio Broadcasting," http://www.worlddab.org

[7] "Digital audio broadcasting," http://en.wikipedia.org/wiki/Digital_Audio_Broadcast

[8] A.J. Bower, "DIGITAL RADIO --The Eureka 147 DAB system," ***Electronic Engineering BBC***, pp.55-56, April 1998.

[9] ETS 300 797, " Radio broadcasting systems; Distribution interfaces; Services Transport Interface (STI)," ETSI, February 1999.

[10] ETS 300 799, " Radio broadcasting systems; Distribution interfaces; Ensemble Transport Interface (ETI)," ETSI, September 1997.

[11] C. Liu, "DAB systems," ***Introduction to Broadcast Technology***, pp. 23-28, 1998.

[12] D. Li, "Discussion on several technologies in DAB," ***Radio & TV Broadcasting Engineering***, pp.78-88, June 1997.

[13] C. Gandy "DAB: an introduction to the Eureka DAB systems and a guide to how it works," ***Research & Development British Broadcasting Corporation***, June 2003.

[14] G. Stoll, "Source coding for DAB and the evaluation of its performance: A major application of the new ISO audio coding standard," ***Proceeding of First International Symposium on Digital Audio Broadcasting***, Montreux, Geneva, 8-9 June 1992.

[15] F. van de Laar, N. Philips and R. Olde Dubbelink, **"**General-purpose and application-specific design of a DAB channel decoder," ***EBU Technical Review*** *no.258*, Winter 1993.

[16] Pommier, R.D., RATLIFF, P.A., and MEIER-ENGELENE, E., "The convergence of satellite and terrestrial system approaches to digital audio broadcasting with mobile and portable receivers," ***EBU Technical Review***, 241/242, pp.82-94, June/August 1990.

[17] David R. Smith, "Digital Transmission Systems," Kluwer Academic, New York, third edition, 2004.

[18] Michael Speth," OFDM Receivers for Broadband-Transmission," May 1999. http://www.ert.rwth-aachen.de/index_e.htm

[19] Weinstein, S.B. and Ebert, P.M., "Data transmission by frequency division multiplexing using the discrete Fourier transform," ***IEEE Transaction on Communications***, vol. COM-19, no.15, pp.628-634, October 1971.

[20] Zou, W. Y. and Wu Y., " COFDM: An Overview**,**" ***IEEE Transaction on Broadcasting***, vol.41, no. 1, March 1995.

[21] Tamaki, Sho and Wada, Tomohisa, "OFDM", Copyright (C) 2001-2005 Magna Design Net, Inc. All Rights Reserved. http://www.magnadesignnet.com/eng/index.html

[22] A. Bruce Carlson, "Communication Systems: An introduction to signals and Noise in Electrical communication," McGraw Hill, Singapore, 1986.

[23] Samuel D. Stearns, "Digital Signal Processing with Examples in MATLAB®," CRC Press LLC, Boca Raton, Florida, 2003.

[24] E. Oran Brigham, "The fast Fourier transforms and its applications," Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[25] ETSI TR 101 496-3," Digital Audio Broadcasting (DAB); Guidelines and rules for implementation and operation; Part 3: Broadcast network", ***EBU Technical Report***, V1.1.2 (2001-05), 2001.

[26] J.H. Stott," Explaining some of the magic of COFDM," ***Proceedings of 20th International Television Symposium***, Montreux, Switzerland, June 1997.

[27] "Software Defined Radio (SDR)" http://dsp.sun.ac.za/wiki/index.php/SDR

[28] Leonard E. Miller and Jhong S. Lee, " BER Expression for Differentially Detected π/4 DQPSK Modulation," *IEEE Transactions on Communications*, vol.46, no.1, pp.71-81, January 1998.

[29] Theodore S. Rappaport, " Wireless Communications: Principal and Practice", Prentice Hall PTR, Upper Saddle River, News Jersey, 1996.

[30] Johannes J. Cronj'e, "Software Design of a Software Defined Radio System," Master thesis, University of Stellenbosch, October 2004.

[31] Les Sabel and Dave Hawkins, " Next Generation Developments in Digital Audio Broadcasting," www.broadcasting.com/radio/BCA03RadioscapeNextGenDAB.pdf

[32] Jeffrey H. Reed, "Software Radio: A Modern Approach to Radio Engineering," Prentice Hall PTR, Upper Saddle River, New Jersey, 2002.

[33] Alok Shah, "An Introduction to Software Radio," Vanu, Inc., 2002. http://www.vanu.com/resources/intro/SWRprimer.pdf

[34] K. Taura et al., "A digital audio broadcasting (DAB) receiver," *IEEE Transactions on Consumer Electronics*, vol.42, no.3, pp.322-326, August 1996.

Appendix A

This appendix discusses the tools used in the thesis. The specifications for both hardware and software used are given.

A.1 Software specification

Linux distribution	SuSE Linux 9.0
Kernel version	2.4.20-4GB-athlon
C++ compiler	gcc version 3.3.1
XML	version 1.0
Windows	Microsoft Windows XP
MATLAB	version 6.5

A.2 Hardware specification

One computer (PC) with DAQ card is used in real time implementation. A computer performs both transmission and reception. Software implementing a transmitter and a receiver are linked by the DAQ card. The DAQ card performs both digital-to-analog and analog-to-digital conversions. The analogue output (I and Q) from the DAC is connected to ADC via two wires.

The simulation is carried out in a laptop computer with the specification given in table below.

Hardware	Specifications			
PC	CPU	AMD Athlon™ XP 1900+ (1600MHz)		
	RAM	256 MB		
DAQ	DAQ-2010	Analog input	Resolution	14 bits, no missing code
			Max sampling rate	2MS/s
			A/D converter	LTC1414
			Number of channels	4 differential
		Analog output	Resolution	12 bits
			Max update rate	1MS/s
			D/A converter	LTC 7545
			Number of channels	2 channels voltage output
Laptop	CPU	Intel® Pentium® M710 processor (1.4GHz)		
	RAM	512 MB		

Table A.2 Hardware specification

A.3 Code

MATLAB, C++ and XML code. A CD attached contains the source code developed.

Appendix B

Phase reference symbol parameter

This appendix discusses the relation between the indices i, k' and n and the carrier index k for the four DAB transmission modes [5]. These provide the parameters that are used in the generation of the phase reference symbol for the respective transmission mode.

Relation between the indices *i*, *k'* and *n* and the carrier index *k* for transmission mode I

k in the range of		*k'*	*i*	*n*
min	max			
-768	-737	-768	0	1
-736	-705	-736	1	2
-704	-673	-704	2	0
-672	-641	-672	3	1
-640	-609	-640	0	3
-608	-577	-608	1	2
-576	-545	-576	2	2
-544	-513	-544	3	3
-512	-481	-512	0	2
-480	-449	-480	1	1
-448	-417	-448	2	2
-416	-385	-416	3	3
-384	-353	-384	0	1
-352	-321	-352	1	2
-320	-289	-320	2	3
-288	-257	-288	3	3
-256	-225	-256	0	2
-224	-193	-224	1	2
-192	-161	-192	2	2
-160	-129	-160	3	1
-128	-97	-128	0	1
-96	-65	-96	1	3
-64	-33	-64	2	1
-32	-1	-32	3	2

k in the range of		*k'*	*i*	*n*
min	max			
1	32	1	0	3
33	64	33	3	1
65	96	65	2	1
97	128	97	1	1
129	160	129	0	2
161	192	161	3	2
193	224	193	2	1
225	256	225	1	0
257	288	257	0	2
289	320	289	3	2
321	352	321	2	3
353	384	353	1	3
385	416	385	0	0
417	448	417	3	2
449	480	449	2	1
481	512	481	1	3
513	544	513	0	3
545	576	545	3	3
577	608	577	2	3
609	640	609	1	0
641	672	641	0	3
673	704	673	3	0
705	736	705	2	1
737	768	737	1	1

Relation between the indices *i*, *k'* and *n* and the carrier index *k* for transmission mode II

k in the range of min	max	*k'*	*i*	*n*
-192	-161	-192	0	2
-160	-129	-160	1	3
-128	-97	-128	2	2
-96	-65	-96	3	2
-64	-33	-64	0	1
-32	-1	-32	1	2

k in the range of min	max	*k'*	*i*	*n*
1	32	1	2	0
33	64	33	1	2
65	96	65	0	2
97	128	97	3	1
129	160	129	2	0
161	192	161	1	3

Relation between the indices *i*, *k'* and *n* and the carrier index *k* for transmission mode III

k in the range of min	max	*k'*	*i*	*n*
-96	-65	-96	0	2
-64	-33	-64	1	3
-32	-1	-32	2	0

k in the range of min	max	*k'*	*i*	*n*
1	32	1	3	2
33	64	33	2	2
65	96	65	1	2

Relation between the indices *i*, *k'* and *n* and the carrier index *k* for transmission mode IV

k in the range of min	max	*k'*	*i*	*n*
-384	-353	-384	0	0
-352	-321	-352	1	1
-320	-289	-320	2	1
-288	-257	-288	3	2
-256	-225	-256	0	2
-224	-193	-224	1	2
-192	-161	-192	2	0
-160	-129	-160	3	3
-128	-97	-128	0	3
-96	-65	-96	1	1
-64	-33	-64	2	3
-32	-1	-32	3	2

k in the range of min	max	*k'*	*i*	*n*
1	32	1	0	0
33	64	33	3	1
65	96	65	2	0
97	128	97	1	2
129	160	129	0	0
161	192	161	3	1
193	224	193	2	2
225	256	225	1	2
257	288	257	0	2
289	320	289	3	1
321	352	321	2	3
353	384	353	1	0

Time-Frequency-Phase parameter *h* values

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$h_{0,.,j}$	0	2	0	0	0	0	1	1	2	0	0	0	2	2	1	1
$h_{1,.,j}$	0	3	2	3	0	1	3	0	2	1	2	3	2	3	3	0
$h_{2,.,j}$	0	0	0	2	0	2	1	3	2	2	0	2	2	0	1	3
$h_{3,.,j}$	0	1	2	1	0	3	3	2	2	3	2	1	2	1	3	2

j	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$h_{0,.,j}$	0	2	0	0	0	0	1	1	2	0	0	0	2	2	1	1
$h_{1,.,j}$	0	3	2	3	0	1	3	0	2	1	2	3	2	3	3	0
$h_{2,.,j}$	0	0	0	2	0	2	1	3	2	2	0	2	2	0	1	3
$h_{3,.,j}$	0	1	2	1	0	3	3	2	2	3	2	1	2	1	3	2

Appendix C

Theoretical BER

This appendix describes the mathematical expression used to determine the theoretical BER for π/4 DQPSK in additive white Gaussian noise (AWGN) based on [28].

Theoretical Derivation:

The theoretical BER for π/4 DQPSK is obtained from the theoretical BER for BPSK. The derivation is according to [28] and illustrated in the following paragraphs:

(a) Transmitted signal:

Consider an OFDM signal generated in frequency domain using the BPSK modulation scheme. The generated signal components have a common magnitude, let's call it A.

Lets' consider a particular component at a specific carrier frequency as illustrated in Figure C1.

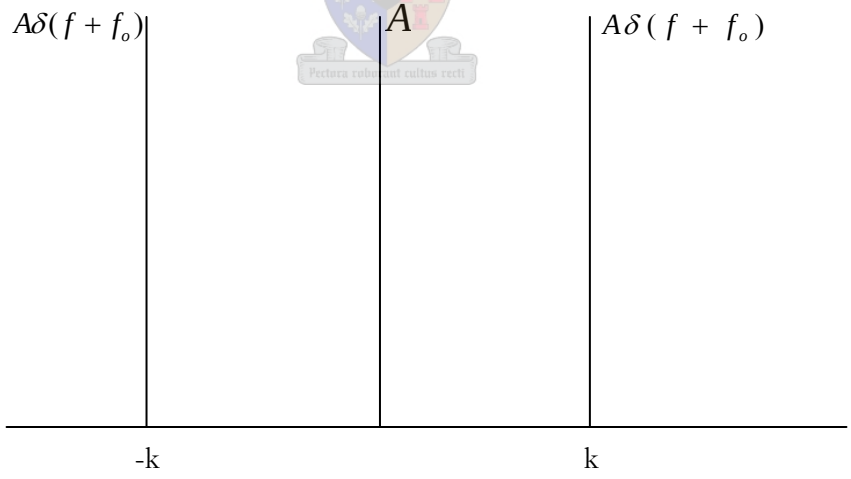


Figure C1 Signal component spectrum.

When an IFFT is applied to signal component, the output time domain signal is presented as shown below:

$$x(t) = \text{IFFT}(X(\omega)) \tag{0.1}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N} \tag{0.2}$$

$$= \frac{A}{N}\left[e^{j2\pi kn/N} + e^{-j2\pi kn/N}\right]$$

$$x(n) = \frac{2A}{N}\cos(2\pi kn/N) \tag{0.3}$$

The IFFT algorithm's output is scaled by a factor N, (i.e. (0.3) is multiplied by N before transmission, see 2.6). The transmitted one BPSK signal is given by:

$$x(n) = 2A\cos(2\pi kn/N) \qquad n = 0,1,...,N-1 \tag{0.4}$$

where N is the number of samples in one symbol period.

(b) Estimating the transmitted signal power using power density spectrum (PDS):

The average signal power in one BPSK signal can be calculated according to [23] as follow:

$$\text{Average signal power} = \frac{1}{N}\sum_{n=0}^{n=N-1}\left[x(n)\right]^2 = \frac{1}{N^2}\sum_{m=0}^{m=N-1}|X(m)|^2 \tag{0.5}$$

$$PDS = \frac{1}{N}|X(m)|^2$$

where (0.6)

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi mn/N} \qquad n = 0,1...N-1$$

Substituting (0.4) into (0.6)

$$X(m) = \sum_{n=0}^{N-1} [2A\cos(2\pi kn/N)[\cos(2\pi mn/N) - j\sin(2\pi mn/N)]]$$
$$= 2A\sum_{n=0}^{N\text{-}1} \cos^2(2\pi mn/N) \qquad for\ m = k\ \ and\ \ m = N - k \tag{0.7}$$
$$= 2A.\frac{N}{2} = AN$$

Hence

$$PDS = \frac{1}{N}(AN)^2 = A^2N \tag{0.8}$$

Then the average signal power in one BPSK signal in terms of PDS is:

$$Avg.power = \frac{1}{N}\sum_{m=0}^{m=N-1} PDS = \frac{1}{N}\sum_{m=k,N-k} A^2N = 2A^2 \tag{0.9}$$

(c) Estimating Noise Power at the receiver:

Noise power is equivalent to its variance (zero mean assumed):

Lets' denote the noise output from DFT as n_{out} and the input noise as n_{in}

$$n_{out} = \sum_{n=0}^{N-1} n_{in}(n)e^{-j2\pi nk/N}$$
$$= \sum_{n=0}^{N\text{-}1} n_{in}(n)\cos(2\pi nk/N) \tag{0.10}$$

The real part of (0.10) was considered.

Lets' define the variance of the noise at the output of DFT var_{out} and from (0.10)

$$
\begin{aligned}
\text{var}_{out} &= E[n_{out}^2] \\
&= E\left[\left[\sum_{n=0}^{N-1} n_{in}(n)\cos(2\pi nk/N)\right]^2\right] \\
&= \sum_{n=0}^{N-1} E\left[n_{in}^2(n)\cos^2(2\pi nk/N)\right] \\
&= N\,\text{var}_{in}/2 \\
&= N\sigma_{in}^2/2
\end{aligned}
\tag{0.11}
$$

(d) Determining the probability of error (BPSK)

The probability density function is defined as:

$$
f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-(x-m)^2/2\sigma^2)} \tag{0.12}
$$

From (0.7) and (0.11), the theoretical probability of error can be calculated as:

$$
P_{e,BPSK} = \frac{1}{\sqrt{\pi N\sigma_{in}^2}} \int_0^{\infty} e^{-\left(\frac{1}{N\sigma_{in}^2}(x+NA)^2\right)} dx
$$

$$
let \quad z = \frac{x+NA}{\sqrt{N\sigma_{in}^2}}
$$

$$
when \qquad x = 0
$$

$$
z = \frac{NA}{\sqrt{N\sigma_{in}^2}} = \sqrt{\frac{A^2N}{\sigma_{in}^2}}
$$

$$
P_{e,BPSK} = \frac{1}{\sqrt{\pi N\sigma_{in}^2}} \cdot \sqrt{N\sigma_{in}^2} \int_{\sqrt{\frac{A^2N}{\sigma_{in}^2}}}^{\infty} e^{-z^2} dz
$$

$$
= 0.5 erfc\left(\sqrt{\frac{A^2N}{\sigma_{in}^2}}\right) \tag{0.13}
$$

Equation (0.13) gives the theoretical expression for determining probability of Bit Error for the BPSK.

[28] shows that π/4 DQPSK is 2.3dB inferior to BPSK. But the OFDM is made of K BPSK carriers. The OFDM signal power with K carriers in one OFDM symbol period can be calculated as follows:

The average signal power is derived from:

$$Avg.OFDM\ \ signal\ power = \frac{1}{N}\sum_{n=0}^{N-1} s_{comp}^2(n)$$

where (0.14)

s_{comp} is an OFDM signal with K carriers in one OFDM symbol period.

Thus one carrier signal average power (BPSK signal) can be given by dividing (0.14) by the number of carrier used in the transmission. From (0.9) average signal power in one BPSK signal is determined.

Relating (0.9) and (0.14) we obtain:

$$2A^2 = \frac{\frac{1}{N}\sum_{n=0}^{N-1} s_{comp.}^2(n)}{K}$$

Hence (0.15)

$$A^2 = \frac{\frac{1}{N}\sum_{n=0}^{N-1} s_{comp.}^2(n)}{2}$$

Substituting (0.15) into (0.13)

$$P_{e,BPSK} = 0.5erfc\left(\sqrt{N.\frac{\frac{1}{N}\sum_{n=0}^{N-1} s_{comp.}^2(n)}{2\sigma_{in}^2}}\right) \quad (0.16)$$

Where

$$Avg.OFDM\ \ signal\ \ power = \frac{1}{N}\sum_{n=0}^{N-1} s_{comp.}^2(n)$$

$$Avg.noise\ power = \sigma_{in}^2$$

K is the total number of carrier used (384 mode II)

Then (0.16) can be re-written as:

$$P_{e,BPSK} = 0.5erfc\left(\sqrt{\frac{N}{K}.\frac{Avg.OFDM\ \ signal\ \ power}{2.Avg.noise\ power}}\right) \tag{0.17}$$

In comparison with Pe for BPSK according to [29],

$$\begin{aligned} P_{e,BPSK} &= Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \\ &= \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \end{aligned} \tag{0.18}$$

Then SNR for BPSK can be calculated from:

$$SNR = \frac{N}{K}.\frac{Avg.OFDM\ \ signal\ \ power}{2.Avg.noise\ power} = \frac{E_b}{N_0} \tag{0.19}$$

[28] describes the relationship between probability of error for BPSK and π/4DQPSK by the following equation:

$$\begin{aligned} P_{e,\pi/4DQPSK} &= Q\left(\sqrt{1.1716.\frac{E_b}{N_0}}\right) \\ &= \frac{1}{2}erfc\left(\sqrt{\frac{1.1716}{2}.\frac{E_b}{N_0}}\right) \end{aligned} \tag{0.20}$$

From [28] and using (0.18) and (0.19), then (0.20) can be re-written as:

$$P_{e,\pi/4DQPSK} = \frac{1}{2}erfc\left(\sqrt{\left(\frac{1.1716}{2}\right).\left(\frac{N}{K}.\frac{Avg.OFDM\ \ signal\ \ power}{2.Avg.noise\ power}\right)}\right) \tag{0.21}$$

Pectora roborant cultus recti