ADAPTIVE SCHEMES FOR MULTICARRIER TRANSMISSION SYSTEM WITH MB-OFDM MODULATION

A THESIS

by

PRIYA HARIHARAN

Submitted to

Communications Laboratory
Faculty of Electrical Engineering and Computer Science
University of Kassel

In Partial fulfillment of the requirements for
M.Sc. degree in Electrical Communications Engineering

March 2007

Supervisors: Dr.sc.techn.Thomas Hunziker
Dipl.-Ing.Mario Westmeier

Examiners: Prof.Dr.sc.techn.Dirk Dahlhaus
Dr.sc.techn.Thomas Hunziker
Declaration

I hereby pledge that this work was carried out abiding the rules and regulations of the Examination committee, Department of Electrical Engineering and Computer science, University of Kassel. Any analogous phrase or sentences from other literature were duly acknowledged.

----------------------------------

Priya Hariharan
Acknowledgements

I would like to express my sincere gratitude to Prof. Dr. sc. techn. Dirk Dahlhaus for offering me this opportunity to work on this thesis. I thank him for all his suggestions and support during my days in his department.

My special thanks to Dr. sc. techn. Thomas Hunziker for his guidance and support throughout my thesis. I thank him for all his valuable suggestions and discussions that I had with him. Thanks for all the detailed explanations he provided in order to clarify my doubts.

I thank Dipl.-Ing. Mario Westmeier for his valuable time and feedback especially on issues related to SystemC. I am thankful to all his suggestions during the debugging phase and on the discussions related to the simulation chain.

I also take this opportunity to thank all the other members of Department of Communications Laboratory.

My deepest thanks to my parents and sister for their constant support, encouragement, understanding & patience throughout my study period.

I thank all my friends here in Germany and back home in India for their motivation and support.
Abstract
This thesis presents the overview on Multi Band OFDM (MB-OFDM) based physical (PHY) layer proposal of WiMedia Alliance for a short range, high data rate Ultra Wide Band (UWB) communications. Further, to improve the BER performance, adaptive power loading is implemented at the transmitter. Adaptive bit loading scheme was attempted, but eventually failed. The implementation difficulties of the adaptive bit loading have been identified. The simulation results and performance analysis are presented.
Contents

List of figures .................................................................................................................. 3

1.0 Introduction .............................................................................................................. 4
  1.1 Motivation .................................................................................................................. 4
  1.2 Contributions ............................................................................................................ 5
  1.3 Outline of this thesis ............................................................................................... 5

2.0 Introduction to OFDM ............................................................................................. 6
  2.1 Principle of OFDM .................................................................................................... 8
  2.2 DFT front end ........................................................................................................... 10
  2.3 Guard Interval (GI) & Cyclic Prefix ........................................................................ 11
  2.4 OFDM processing .................................................................................................... 12
  2.5 Performance Improvement of OFDM ...................................................................... 12
    2.5.1 Coded OFDM (COFDM) ..................................................................................... 12
    2.5.2 Bit Interleaved Coded Modulation (BICM)-OFDM ........................................... 13
    2.5.3 Adaptive Schemes ............................................................................................. 13
  2.6 Advantages of OFDM .............................................................................................. 14
  2.7 Disadvantages of OFDM ......................................................................................... 15

3.0 Ultra Wide Band (UWB) .......................................................................................... 16
  3.1 Single Band ............................................................................................................. 19
    3.1.1 Impulse Radio .................................................................................................... 19
  3.2 MultiBand ................................................................................................................ 19
    3.2.1 Pulsed Multiband UWB .................................................................................... 20
    3.2.2 Multiband based on OFDM .............................................................................. 20
  3.3 UWB Channel Model ............................................................................................... 21

4.0 Multi Band OFDM Alliance (MBOA) System ...................................................... 23
  4.1 MBOA transmitter architecture .............................................................................. 25
    4.1.1 Data Scrambler ................................................................................................. 25
    4.1.2 Convolution Encoder ....................................................................................... 26
    4.1.3 Puncturing ........................................................................................................ 27
    4.1.4 Interleaver ......................................................................................................... 27
    4.1.5 Constellation Mapping ..................................................................................... 29
      4.1.5.a QPSK .......................................................................................................... 30
      4.1.5.b Dual Carrier Modulation ............................................................................ 31
    4.1.6 Spreading ......................................................................................................... 33
List of figures

Figure 2.1 Comparison of SCM and MCM ................................................................. 6
Figure 2.2 Comparison of transmitted waveforms and frequency spectra. ................. 9
Figure 2.3 Spectra of 3 different OFDM subcarriers .................................................. 10
Figure 2.4 Signal in time domain along with guard interval ........................................ 10
Figure 2.5 Transmitted and Received signals a) no guard interval; b) with guard interval; c) guard interval with cyclic prefix ................................................................. 11
Figure 2.6 OFDM processing ...................................................................................... 12
Figure 2.6 OFDM – robustness to multipath propagation .............................................. 14
Figure 3.1 UWB and narrowband spectral occupancy .................................................. 16
Figure 3.2 UWB versus other radio communication systems ....................................... 17
Figure 3.3 Emission spectrum mask ............................................................................ 17
Figure 3.4 Impulse radio – signals in time and frequency domain................................. 19
Figure 3.5 Pulsed multiband UWB ............................................................................. 20
Figure 3.6 Multiband OFDM ...................................................................................... 21
Figure 3.7 UWB indoor channel impulse response characteristics ................................ 22
Figure 4.1 MBOA band allocation .............................................................................. 23
Figure 4.2 Multi Band OFDM ..................................................................................... 24
Figure 4.3 MBOA transmitter architecture ................................................................. 25
Figure 4.4 Data scrambler ......................................................................................... 26
Figure 4.5 Convolution Encoder ................................................................................ 26
Figure 4.6 Puncturing for ½ rate encoder ................................................................... 27
Figure 4.7 Interleavers ............................................................................................... 28
Figure 4.8 3 stages of Interleaver for 53 Mbps ............................................................. 29
Figure 4.9 QPSK constellation mapping ..................................................................... 30
Figure 4.10 Dual carrier modulation .......................................................................... 31
Figure 4.11 Constellation diagram for DCM ............................................................... 32
Figure 4.12 Structure of 1 OFDM symbol ................................................................. 34
Figure 4.13 FFT ......................................................................................................... 34
Figure 4.13 Logical Subcarriers ................................................................................ 35
Figure 5.1 Transmitter architecture with adaptive power loading block ...................... 37
Figure 5.2 Illustration of water filling principle [1] ...................................................... 38
Figure 5.3 Water Filling coefficients for 16-QAM and SNR=6dB ................................. 39
Figure 5.4 Water Filling coefficients for 16-QAM and SNR=13dB [2] ......................... 39
Figure 5.5 Water filling principle – channel capacities [2] .......................................... 40
Figure 5.6 Binary symmetric channel ......................................................................... 41
Figure 5.7 Lambert’s W function [2] ......................................................................... 43
Figure 5.8 Adaptive power loading algorithm ............................................................ 44
Figure 5.9 Adaptive power loading for QPSK, 200 Mbps, Eb/No 3 dB ....................... 45
Figure 5.10 Adaptive power loading for QPSK, 200 Mbps, Eb/No 6 dB ..................... 45
Figure 5.11 Adaptive power loading for QPSK, 200 Mbps, Eb/No 9 dB ...................... 46
Figure 5.12 Adaptive power loading algorithm when adaptive bit loading is employed 48
Figure 6.1 SystemC architecture .............................................................................. 49
Figure 6.2 Entities of “TestBench” module ................................................................. 52
Figure 6.3 Entities of transmitter “TxH2” module ....................................................... 53
Figure 6.4 Entities of channel “ChannelH2” module .................................................. 54
Figure 6.5 Entities of receiver “RxH2” module ........................................................... 54
Figure 6.6 Transmitter module with adaptive power loading module ......................... 55
Figure 6.7 Frame structure ....................................................................................... 56
Figure 6.8 Illustration of temporary channel and channel coefficients flow .................. 57
Figure 6.9 “TestBench” module with temporary channel module “TempChannel” .... 58
Figure 6.10 Illustration - sending APL coefficients to receiver .................................... 59
Figure 6.11 Adaptive bit loading algorithm ............................................................... 61
Figure 6.12 “RxH2” with reverse adaptive bit loading module “TempAdaptivePowerLoading” 62
Figure 6.13 Adaptive Power Loading algorithm for same channel conditions and different Eb/No values 64
Figure 6.14 200 Mbps, NLOS channel (CM2), with MRC, BER and PER performances 65
Figure 6.15 200 Mbps, NLOS channel (CM2), with MMSE, BER and PER performances 66
Figure 6.16 480 Mbps, NLOS channel (CM2), with MMSE, BER and PER performances 67
Figure 6.17 53.3 Mbps, NLOS channel (CM2), with MRC, BER and PER performances 68
Figure 6.18 53.3 Mbps, NLOS channel (CM2), with MMSE, BER and PER performances 69
1.0 Introduction

WiMedia is the standard radio platform for Ultra Wide Band (UWB) wireless networking. The standard incorporates Medium Access Control (MAC) and physical (PHY) layer specifications based on Multi Band – Orthogonal Frequency Division Multiplexing (MB-OFDM) technique. It offers solution to enable data rates upto 480 Mbps with lower power consumption. The platform is also optimised for Wireless Personal Area Network (WPAN) technology such as Bluetooth 3.0, Wireless Universal Serial Bus (USB). The standard utilises the unlicensed 3.1 to 10.6 GHz UWB and divides the spectrum into 14 bands each with bandwidth of 528 MHz.

At the physical layer, power, modulation and other parameters can be dynamically adapted to changing channel conditions. By this way, the channel can be utilised more efficiently.

1.1 Motivation

Many broadband wireless transmission systems employ Orthogonal Frequency Division Multiplexing (OFDM) as the modulation scheme due to simple resulting transceiver structure. The complexity is a major issue if handheld battery powered devices are used, where a trade-off has to be found between achievable information bandwidth and power consumption of the signal processing in the corresponding terminals.

WiMedia-Multi Band OFDM Alliance (MBOA) is a platform for Ultra Wide Band (UWB) wireless networking offering high speed wireless connection for next generation consumer electronics, mobile terminals and computer applications.

One way to improve the system capabilities of an OFDM transceiver in slowly fading radio channels results from exploiting the Channel State Information (CSI) at the transmitter in order to implement different adaptive schemes. The adaptive schemes are based on the minimization of bit-error rate at the decoder output under constraints on the available power and required data rate. In the present project, the schemes are to be applied to a modified OFDM system, where a multi band OFDM modulation technique is adopted. To this end, suitable modifications of the existing schemes should be elaborated and analyzed. The found schemes are to be implemented and analyzed in an existing SystemC simulation environment.
1.2 Contributions

The basic aim of this thesis is to implement the adaptive schemes in the MB-OFDM system. The following lists the main contribution of this thesis

- Adaptive power loading was implemented with hard decision decoding at the receiver
- An attempt was made to implement adaptive bit loading. The scheme was not successful and the problems were identified
- Simulation results for various scenarios were performed and the BER performance was analyzed

1.3 Outline of this thesis

Chapter 2 gives a brief introduction on multicarrier transmission and principle of OFDM.
Chapter 3 gives an overview on UWB, multi band OFDM and UWB channel modelling for indoor applications
Chapter 4 presents the MBOA system architecture, frequency sub banding.
Chapter 5 discusses on the adaptive schemes with main focus on adaptive power loading and adaptive bit loading
Chapter 6 gives an introduction to MBOA simulator, description of the modules, implementation of adaptive power loading and bit loading. This chapter also includes the simulation results.
2.0 Introduction to OFDM

OFDM is a special form of Multi Carrier Modulation (MCM), where a single data stream is transmitted over a number of lower rate subcarriers. Each subcarrier is orthogonal to every other subcarrier. OFDM increases the robustness against frequency selective fading and narrowband interference. In a single carrier system, a single fade or interfere can cause the entire link to fail, but in multicarrier system, only a small percentage of subcarriers will be affected.

MCM is the principle of transmitting data by dividing input stream into several symbol streams, each of which has a much lower symbol rate, and by using these sub streams to modulate several subcarriers.

Figure 2.1 compares a single carrier modulation (SCM) and an multicarrier modulation (MCM).

![Figure 2.1 Comparison of SCM and MCM][4]
Bandwidth of the SCM is denoted by $B_{SCM}$ and bandwidth of MCM is denoted by $B_{MCM}$. The amplitude spectrum of the MCM signal is written as,

$$S_{MCM}(f) = \sum_{k=1}^{N_{sc}} F_k(f), \quad (2.1)$$

Where, $N_{sc}$ denotes the number of subcarriers, $F_k(f)$ denotes the amplitude spectrum of pulsed waveform of the $k^{th}$ subcarrier.

Let the transfer function of a frequency selective fading channel be $H(f)$, then the amplitude spectra of received single carrier and multicarrier signals are written as,

$$R_{SCM}(f) = H(f)S_{SCM}(f), \quad (2.2)$$

$$R_{MCM}(f) = H(f)S_{MCM}(f), \quad (2.3)$$

In the equation (2.2), (2.3) $R_{SCM}(f)$ & $S_{SCM}(f)$ represents the amplitude spectrum of received and transmitted single carrier modulation signal, $R_{MCM}(f)$ & $S_{MCM}(f)$ represents the received and transmitted multicarrier modulation signal. Further, $R_{MCM}(f)$ can be written as,

$$R_{MCM}(f) = \sum_{k=1}^{N_{sc}} H_k(f)F_k(f), \quad (2.4)$$

where $H_k(f)$ is a transfer function of a frequency sub band $B_k$ occupied by the $k^{th}$ subcarrier in a multicarrier system. If the number of subcarrier are relatively large, then the amplitude and phase response of $H_k(f)$ over $B_k$ can be assumed to be constant.

This shows that a MCM system has robustness against frequency selective fading. Also, MCM requires utmost a single tap equaliser whereas SCM requires adaptive equalisation.

OFDM can be thought of modulation that allows multiple users to share the communication channel. OFDM segments the channel according to frequency. Basically, it divides the available spectrum into a number of equally spaced tones. Each tone carries a portion of
user’s information. One of the important properties of OFDM is that each tone is orthogonal to every other tone. Since the tones are orthogonal, OFDM allows these tones to overlap. By this way, spectrum is efficiently used.

The sinusoidal waveforms making up the tones in OFDM have a very special property of being the only Eigen-functions of the sub channel. This is the reason they are orthogonal even if the tones are overlapped. The concept of orthogonal property of OFDM signal is explained in further sections.

### 2.1 Principle of OFDM

The OFDM transmitted signal $s(t)$ is of the form,

$$s(t) = \sum_{i=-\infty}^{\infty} \sum_{k=1}^{N_s} c_{ki} e^{j2\pi k_i(t-iT_s)} f(t-iT_s),$$

(2.6)

where, $T_s$ denotes the subcarrier symbol period, $c_{ki}$ is the $i$th information bit on the $k$th subcarrier, and $f(t)$ is the pulsed waveform of the symbol.

If a rectangular pulse is used, then

$$f(t) = \begin{cases} 1, & 0 < t \leq T_s \\ 0, & t \leq 0, t > T_s \end{cases}.$$  

(2.7)

The comparison of baseband transmission system and OFDM system is shown in Figure 2.2 (a) and 2.2 (b)

Figure 2.2 (c), (d), (e), (f) shows the frequency spectra in both cases.
It can be seen in Figure 2.2 that classical MCM such as impulse radio employing non-overlapping orthogonal signals though requires larger bandwidth, can be implemented using the analog oscillators and filters. If a rectangular pulse waveforms are used for the subcarriers, the frequency spectra of the waveform is widely spread and overlapped. By this way, spectral efficiency is achieved. But however analog filters introduces difficulty in subcarrier recovery at the receiver without inter subcarrier interference. Due to this fact, OFDM was initially abandoned.
2.2 DFT front end

Parallel data transmission was achieved by the concept of using DFT as a part of digital modulation/demodulation. Using DFT, the subcarriers are shaped like \( \sin(x)/x \). Figure 2.3 shows the spectra of 3 different OFDM subcarriers.

![Spectra of 3 different OFDM subcarriers.][8]

It can be seen that 3 different spectra are partially overlapping and hence increased spectral efficiency is achieved compared to that of conventional non-overlapped multicarrier system. The same OFDM signal is shown in time domain in Figure 2.4.

![Signal in time domain along with guard interval.][8]

The use of IDFT/DFT totally eliminates the bank of subcarrier oscillators at the transmitter/receiver end. If the number of subcarriers is chosen as a power of 2, then DFT can be efficiently implemented by Fast Fourier Transform (FFT). It is clear that the sinusoidal tones are the only waveform that ensures orthogonality between OFDM signals.
2.3 Guard Interval (GI) & Cyclic Prefix

In a multipath propagation channel, to increase the robustness of OFDM against Inter Symbol Interference (ISI), an addition of guard period and cyclic prefix is introduced.

Consider a subcarrier $k$, and 2 paths of the transmitted signal at the receiver. Let us assume that the 1st path is the desired signal and 2nd path is the delayed signal. This introduces a distortion of 2 signals at the receiver as shown in Figure 2.5(a). This can be eliminated by extending a guard interval, where no signal is transmitted. By this way, the ISI can be perfectly eliminated. This guard period should be greater than the channel delay period. The transmitted and received signal with guard intervals is shown in Figure 2.5(b).

Guard Interval along with cyclic prefix perfectly eliminates the intersubcarrier interference. Here, the symbol length is prolonged with a guard interval, which prefixes a copy of last N samples at the start of OFDM symbol as shown in Figure 2.5(c).
However, it is clear that insertion of this guard interval, decreases the channel capacity. Hence, it is important to choose the guard interval effectively in order to achieve a trade off between the ISI and data rate.

2.4 OFDM processing

The basic OFDM transmitter-receiver structure of OFDM is shown in Figure 2.6. The GTX blocks does the baseband processing – interleaving, channel coding, mapping raw bits to complex symbols. After mapping the complex symbols are sent to a serial to parallel converter (DEMUX) followed by IDFT which converts the signal to time domain. The other functionality of IDFT has been discussed in previous section. The OFDM symbol is added with GI & cyclic prefix. The analog signal is then send through a channel.

The receiver does the reverse of the transmitter functions. In addition a GRX includes a equaliser to restore the amplitude and phase distortions caused due to channel effects.

![Figure 2.6 OFDM processing][8]

2.5 Performance Improvement of OFDM

There are several ways of improving the performance of OFDM in a fading channel. 3 such ways are discussed in the following sections.

2.5.1 Coded OFDM (COFDM)

One such way to improve the performance of OFDM, is to employ channel coding to yield coded OFDM. Coding provides efficient means of obtaining diversity on a fading channel. The amount of diversity produced by a code is directly related to its hamming distance.
Time diversity is achieved by transmitting the same information in multiple time intervals mutually separated by an amount equal to or greater than the coherence time \( (\Delta t)_c \). In a similar way, frequency diversity is achieved by transmitting the same information in multiple frequency slots mutually separated by an amount equal to or greater than the coherence bandwidth \( (\Delta f)_c \) of the channel. By this way, signal components carrying information achieves statistically independent fading.

In coded OFDM, the bit error probability is lowered at the receiver by encoding the bits to a code word. One such encoding scheme is via convolution codes. A convolution code is generated by passing the information sequence through a linear finite state shift register. A shift register consists of \( K \) stages and \( N \) generator functions. If the output number of bits for each \( k \)-bit sequence is \( n \), then the code rate is defined as \( R_c = k/n \). It is shown in [2] that Coded OFDM system has a improved BER-performance over uncoded – OFDM systems.

### 2.5.2 Bit Interleaved Coded Modulation (BICM)-OFDM

An interleaver is a device that re-arranges the order of the input sequence. The inverse of the process is de-interleaving which is carried out at the receiver. This means, the adjacent error bits are separated and hence correlation caused by the burst errors are cancelled.

It has been recognised in [6] that performance of COFDM can be improved by bit wise interleaving the code word and by using a soft decision Viterbi decoder at the receiver. In BICM-OFDM, the encoded bits are further bit-wise interleaved via an ideal interleaver and then mapped to gray labelled constellation points. At the output, the receiver demodulates and de-interleaves the received symbols. Codewords are then decoded using an iterative decoder.

It is shown in [6] that in Rayleigh fading channel, the outage capacity of BICM-OFDM beats that of the COFDM for all values of SNR.

### 2.5.3 Adaptive Schemes

Another way to improve the performance of OFDM system in a slow fading channel conditions is by employing adaptive schemes. In this approach an estimate of the channel is
made available to the transmitter. Several adaptive techniques such as adaptive power loading, Adaptive modulation / bit loading, adaptive coding may be employed. By this way, favourable channel conditions are exploited.

[1] employs adaptive power loading to a slow fading, time invariant channel with an objective of minimising the BER at the decoder output.

[2] shows further adaptive schemes such as adaptive bit loading and adaptive coding.

The adaptive schemes are discussed in detail in Chapter 5.

2.6 Advantages of OFDM

The advantages of OFDM are summarised below

- OFDM is spectrally efficient – As described in the previous sections, IDFT/IFFT generates orthogonal signals that ensures that the sub carriers do not interfere with each other when they are overlapped
- Robustness against narrowband interference
- Robustness to multipath propagation - virtually all paths received within the guard period are captured. Figure 2.6 shows OFDM’s robustness to multipath
- Simple equalization techniques

![OFDM Symbol](image)

**Figure 2.6** OFDM – robustness to multipath propagation


2.7 Disadvantages of OFDM

- Sensitive to Doppler shift – For OFDM, the sub carrier frequency at the transmitter and receiver should be synchronised. When the source and receiver are relatively moving, frequency at the receiver will no longer be the same as the transmitter. This amount of frequency changes because of Doppler effect depends on the relative motion between the source and the receiver. Since OFDM is sensitive to carrier frequency, this may cause serious Inter Carrier Interference (ICI) which may worsen the BER performance & carrier synchronisation errors.
- Sensitive to frequency synchronisation problems
- The linear power amplifier requirement results in inefficient transmission power consumption
3.0 Ultra Wide Band (UWB)

Ultra Wide Band is the radio communications technology that uses a very large bandwidth of 500 MHz or more to transmit information at high speed and relatively short range distances. Federal Communication Commission defines UWB as any signal occupying more than 500 MHz in the 3.1 to 10.6 GHz spectral band or having a fractional bandwidth greater than 20 percent. Fractional bandwidth is given by,

\[ \frac{f_H - f_L}{f_c}, \]

where, \( f_H \) is the higher frequency, \( f_L \) is the lower frequency, \( f_c \) is the centre frequency.

Figure 3.1 shows the spectral occupancy of narrow band and UWB signals. As it can be seen, the fractional bandwidth of narrow band signal is less than 1%.

UWB uses extremely short pulses to carry information. This results in energy spectrum of wide range and hence a low power spectral density.
Figure 3.2 shows the co-existence of radio communication systems technology. Since the power spectrum of UWB devices overlaps with various other technologies, transmission power is restricted by a spectral mask released by FCC. Thus interference of UWB with narrow band systems is avoided by strict regulation of power levels.

Figure 3.3 shows the emission spectrum mask for indoor applications. It can be noted that permitted emission level within the allocated band is $-41.25\text{dBm/MHz}$. 

\[17\]
Shannon’s channel capacity theorem is given by,

\[
C = B \log \left( 1 + \frac{S}{N} \right),
\]

(3.2)

Shannon’s channel capacity theorem states that capacity of a channel increases linearly with bandwidth and increases logarithmically with transmitted power. Thus UWB system which operates on larger bandwidth can achieve higher data rates with low transmission power.

The advantages of UWB are:

- Very high data rates over short range distances which interests High speed Personal Area Networks (PANs)
- Very low transmission energy which results in good battery life and minimal interference with other systems
- Good multipath immunity
- Ability to penetrate walls and other obstructions
- Low cost with minimal RF electronics

UWB systems has a broad range of applications in different areas such as:

- Wireless Communication Systems – Local Area Networks (LANs) and Personal Area Networks (PANs)
- Radar and sensing, etc.
- Wireless Personal Area Networks (WPAN) – interconnecting PAN devices such as mobile handsets, PDAs, laptops, cameras, MP3 players, etc offering much higher data rates than conventional bluetooth or 802.11.

With the available bandwidth of 7.5 GHz, and with minimum signal bandwidth of 500 MHz, UWB systems can be divided into 2 groups:

- Single Band
  - Impulse Radio
Multi Band
  o Pulsed Multiband
  o Multiband based on OFDM

3.1 Single Band
A single band system is a traditional approach to UWB. Impulse radio is based on single band systems.

3.1.1 Impulse Radio
One of the implementation of UWB systems is impulse radio. The concept of impulse radio is to transmit pulses of extremely short durations (sub nanoseconds) rather than a continuous waves to transmit information.

The pulse directly generates a wide bandwidth signal according to time scaling properties of fourier transform relationship between time and frequency.

3.2 MultiBand
A multiband system divides the bandwidth into several smaller sub bands and transmits information across each sub band. Figure 3.5 shows the signal spectrum of a multi band UWB system. According to modulation schemes on each sub band, multi band system can further be divided into two types

  o Pulsed multiband UWB
3.2.1. Pulsed Multiband UWB

A pulsed multiband uses single carrier on each band to transmit information simultaneously. The concept of pulsed multiband UWB is illustrated in Figure 3.5.

By applying pulsed multiband UWB in a multipath fading channel, performance improvement can be achieved better than in the case of single band.

3.2.2. Multiband based on OFDM

In this type of system, the transmitted information over each symbol is modulated using Orthogonal Frequency Division Multiplexing (OFDM) techniques. Figure 3.6 shows the concept of OFDM.

OFDM has several advantages over pulsed multiband such as spectral efficiency and immunity to multipath effects. The concepts and further features of MB-OFDM shall be explained in Chapter 4.
3.3 UWB Channel Model

In order to evaluate different PHY layer proposals, IEEE 802.15.3a channel modelling committee proposed a channel model for realistic UWB environments. Basically, 4 indoor channels are specified namely CM1, CM2, CM3, CM4. Out of the 4 channels, CM1 and CM3 covers the LOS environments, CM2 and CM4 covers the NLOS environments.

The Channel model characteristics are summarised below:

<table>
<thead>
<tr>
<th>Channel</th>
<th>CM1</th>
<th>CM2</th>
<th>CM3</th>
<th>CM4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean excess delay (ns)</td>
<td>5.05</td>
<td>10.38</td>
<td>14.18</td>
<td>-</td>
</tr>
<tr>
<td>RMS delay (ns)</td>
<td>5.28</td>
<td>8.03</td>
<td>14.28</td>
<td>25</td>
</tr>
<tr>
<td>Distance (m)</td>
<td>0-4</td>
<td>0-4</td>
<td>4-10</td>
<td>10</td>
</tr>
<tr>
<td>Scenario</td>
<td>LOS</td>
<td>NLOS</td>
<td>LOS</td>
<td>NLOS</td>
</tr>
<tr>
<td>No.of significant paths (85%)</td>
<td>20.8</td>
<td>33.9</td>
<td>64.7</td>
<td>123.3</td>
</tr>
</tbody>
</table>

Table 3.1 Channel model characteristics of UWB. [14]
Below, Figure 3.2 shows the channel response of the corresponding 4 channels.

**Figure 3.7** UWB indoor channel impulse response characteristics[3]
4.0 Multi Band OFDM Alliance (MBOA) System

MBOA system utilizes the unlicensed band of 3.1 to 10.6 GHz UWB. It offers high speed short range wireless network supporting data rate upto 480 Mbps. The available bandwidth is divided into 14 band groups with 528 MHz bandwidth each as shown in Figure 4.1. A multiband OFDM scheme is used to transmit information. There are a total of 128 subcarriers out of which 100 are data subcarriers.

![Figure 4.1 MBOA band allocation.][3]

It can be clearly understood that transmitter and receiver process signals with smaller bandwidth and this reduces the overall complexity of the transceiver architecture. The specifications of MBOA system is summarised in Table 4.1 below,

![Table 4.1 MBOA specifications.][3]

The structure of the frame and OFDM symbol are explained in the later sections.
Currently, the simulator works in Band group 1 with 3 sub bands. OFDM symbols within a frame are transmitted in one of these sub bands determined by the time frequency code (TFC).

Figure 4.2 shows the hopping of OFDM symbols across each band and Table 4.2 summarises the TFC codes. The advantages of multi banding, OFDM transmission across different sub bands, guard interval and zero prefixes are

- Frequency diversity
- Robustness to multipath
- Transmitter and receiver settling times

![Figure 4.2 Multi Band OFDM][3]

It can be noticed in Table 4.2 that the standard supports Fixed Frequency Interleaving (FFI), in addition to Time Frequency Interleaving (TFI). It is equivalent to transmitting in a single frequency band similar to Frequency Division Multiple Access (FDMA).

<table>
<thead>
<tr>
<th>TFC Number</th>
<th>Type</th>
<th>Preamble</th>
<th>BAND_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TFI</td>
<td>1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>2</td>
<td>TFI</td>
<td>2</td>
<td>1 3 2 1</td>
</tr>
<tr>
<td>3</td>
<td>TFI</td>
<td>3</td>
<td>1 1 2 2</td>
</tr>
<tr>
<td>4</td>
<td>TFI</td>
<td>4</td>
<td>1 1 3 3</td>
</tr>
<tr>
<td>5</td>
<td>FFI</td>
<td>5</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>6</td>
<td>FFI</td>
<td>6</td>
<td>2 2 2 2</td>
</tr>
<tr>
<td>7</td>
<td>FFI</td>
<td>7</td>
<td>3 3 3 3</td>
</tr>
</tbody>
</table>

**Table 4.2 Time Frequency Codes (TFC).**
Some of the notable features in MB-OFDM are

- Fixed Frequency Interleaving in addition to Time Frequency Code
- Dual Carrier Modulation (DCM) for higher data rates.
- 3 stage interleaver at symbol and tone level with a cyclic shifter

4.1 MBOA transmitter architecture

Figure 4.3 shows the MBOA transmitter architecture.

![MBOA transmitter architecture](image)

4.1.1 Data Scrambler

A scrambler randomizes the input sequences without removing any undesirable sequences. It is also known as a randomizer.

The polynomial generator $g(D)$ is defined by

$$g(D) = 1 + D^{14} + D^{15},$$  \hspace{1cm} (4.1)

$D$, represents a single delay element.
Using this generator function, the output \( x(n) \) can be computed as,

\[
x[n] = x[n-14] \oplus x[n-15] \quad n = 0, 1, 2, \ldots ,
\]

and \( \oplus \) represents the modulo-2 addition. If \( s[n] \) represents the input data bits and \( v[n] \) represents the scrambled output, then

\[
v[n] = s[n] \oplus x[n], \quad m = 0, 1, 2, \ldots ,
\]

### 4.1.2 Convolution Encoder

A convolution encoder generates a code by passing the information sequence through a linear finite state shift register. Here, the encoder employs \( R = 1/3 \) rate with generator polynomials

\[
g_0 = 133_8, \quad g_1 = 165_8, \quad g_2 = 171_8.
\]
As shown in Figure 4.5 each input bit has 3 corresponding output bits. Bit ‘A’ is generated 1st followed by bit ‘B’ and bit ‘C’.

### 4.1.3 Puncturing

Other coding rates are derived by puncturing the mother encoded bits. Puncturing is a process of omitting some of bits in order to increase the coding rate. The omitted bits at the transmitter are replaced by 0’s. The process of puncturing is illustrated in the Figure 4.6 Further puncturing process for higher data rates are illustrated in [3].

![Figure 4.6 Puncturing for ½ rate encoder.](image)

### 4.1.4 Interleaver

The coded bits are passed through an interleaver. Here the coded bits are interleaved prior to mapping to increase the diversity order. Interleaving is a process of rearranging the input bits
to provide robustness against burst errors. At the receiver, the inverse process termed as de-interleaving is done to restore the original bits. In a deep fading channel, bit errors occur in burst rather than randomly scattered errors. An effective way to overcome this is by interleaving the input bit sequence such that burst errors are transformed into linearly independent errors. By this way, adjacent errors are separated and results in cancellation of burst errors.

In MBOA, the bit interleaving operation is done in 3 distinct stages to attain frequency diversity within a band group and within a subcarrier.

![Interleavers](image)

**Figure 4.7** Interleavers.

As shown in the Figure 4.7 the 3 stages are

- **Symbol Interleaver**: Here, the input bits are read-in column by column and the output bits are read-out row by row. Thus it permutes the bits across 6 consecutive OFDM symbols. By this way frequency diversity within a band group is achieved.

- **Tone Interleaver**: Permutes the bits across the data subcarriers of each OFDM symbol. It provides robustness against narrow band interferences.
- **Cyclic Shifter**: Cyclically shifts the bits in each OFDM symbol to exploit better frequency diversity.

![Diagram: 3 stages of Interleaver for 53 Mbps.][3]

### 4.1.5 Constellation Mapping

For lower data rates from 53 Mbps to 200 Mbps, the encoded, interleaved input bits shall be mapped to QPSK constellation. For data rates higher than 200 Mbps, input bits shall be mapped based on Dual Carrier Modulation (DCM) technique.
4.1.5.a QPSK

A Gray labelled constellation mapping is chosen as show in Figure 4.9. The normalisation factor of QPSK mapping is $K_{\text{MOD}} = 1/\sqrt{2}$.

**Figure 4.9** QPSK constellation mapping.
4.1.5.b Dual Carrier Modulation

One of the key features of WiMedia’s Multiband – OFDM is the concept of Dual Carrier Modulation. From the MBOA system parameter specification, it can be noticed that spreading rate is varied in order to achieve different data rates.

For data rates under 106.7 Mbps, both frequency and time spreading is done. This not only offers lower data rate but also frequency and time diversity. For data rates from 106.7 Mbps – 200 Mbps, only time spreading is specified. This ensures diversity in time domain.

Whereas for data rates above 320 Mbps, no spreading is done in order to achieve higher these data rates. In these scenarios where there is deep fade, the system has to solely rely on the strength of the error correction codes to recover the lost information. But as the strength of error correction code decreases, there is loss in diversity resulting in performance degradation.

In order to make up for this loss in diversity WiMedia adopts the concept of Dual Carrier Modulation. The concept is shown in Figure 4.10.

It can be seen that 4 interleaved bits are mapped into 2 different constellation Mapping with different Gray code labelling. This yields a 16-QAM like constellation. The 2 different constellation diagrams are shown in Figure 3.11.

Figure 4.10. Dual carrier modulation.
Also, these tones are separated by 50 tones at the IFFT input. The advantages of DCM are

- The same 4 bits are mapped into 2 different constellation mapping
- The probability that the symbols suffer deep fade on both tones is less
Even if one tone suffers deep fade, the information can still be recovered using simple error detection schemes. Loss due to diversity is reduced without time and frequency domain spreading.

4.1.6 Spreading

WiMedia – MBOA offers 2 types of spreading. The main purposes of spreading are:

- To provide time and/or frequency domain diversity
- To improve the performance in the presence of other non-coordinated devices
- To provide variable data rates (aided by puncturing)

In time domain spreading, the same sample is sent twice in 2 different OFDM symbols. This is implemented to achieve data rates of 106.7 and 200 Mbps with different puncturing rate in both the cases. This provides time domain diversity and reduces the probability of error at the output.

For data rates less than 106.7 Mbps both frequency and time domain spreading is done, which means the spreading rate is 4. Time domain spreading is same as explained before. To achieve frequency domain diversity, the same tone is transmitted twice within the same OFDM symbol, one being the complex conjugate of the actual tone. By this way, both time and frequency diversity is achieved.

4.1.7 OFDM Modulation

The structure of 1 OFDM symbol is shown in Figure 4.12. One way to implement the Inverse Discrete Fourier Transform (IDFT) is by using a Inverse Fast Fourier Transform (IFFT). The logical frequency subcarrier is shown in Figure 4.14.
Figure 4.12 Structure of 1 OFDM symbol.

Figure 4.13 FFT.[3]
The logical frequency subcarrier 1 to 61 are mapped to IFFT inputs of 1 to 61. While logical frequency subcarrier of –61 to –1 are mapped to IFFT inputs of 67 to 127.

For data rates 53.3 Mbps and 80 Mbps, a set of 50 complex numbers are grouped. Within an OFDM symbol, these symbols are transmitted twice, one being the complex conjugate of other. For data rates of 106.7 Mbps and 200 Mbps, a set of 100 complex numbers are grouped. These symbols are transmitted twice in 2 different OFDM symbols, mapped on to the same subcarrier.
5.0 Adaptive Schemes

The adaptive schemes are based on the objective of improving the BER performance at the output of the decoder. Two such adaptive schemes are discussed in this chapter – adaptive power loading and adaptive bit loading. Under adaptive power loading, water filling principle which aims at maximising the subchannel capacity and optimised power control algorithm to minimise BER are discussed.

5.1 Power Control Techniques

In high speed wireless data communication systems, it is essential to have robust and spectrally efficient communication techniques in a fading channel. If the channel can be estimated and sent back to the transmitter, various adaptation techniques can be employed relative to the channel characteristics. This allows the channel to be used more efficiently by exploiting the favourable channel conditions.

\textit{Shannon’s Theorem}

Shannon’s Theorem establishes a bound on the theoretical maximum error free data that can be transmitted over a noisy channel with a specified bandwidth $B$. The channel capacity is expressed as

$$C = B \log \left( 1 + \frac{S}{N} \right), \quad (5.1)$$

Where $C$ is the channel capacity, $B$ is the specified bandwidth, $S$ is the average transmitted signal power and $N$ denotes the additive white Gaussian noise power.

If the information is transmitted at the rate $R$ and if,

$$R < C, \quad (5.2)$$
there exist coding techniques which result in minimum probability of error at the receiver. This means, it is possible to transmit the information over a noisy channel without error close to $C$ bits/sec.

On the converse, if,

$$ R > C, $$

(5.3)

then no useful information can be transmitted beyond channel capacity. Figure 4.3 is modified to Figure 5.1 as a consequence of adaptive power loading.

### 5.1.1 Water Filling Principle

OFDM transmit signal is assumed to have the form

$$ s(t) = \sum_{n=1}^{N_b} \sum_{k=1}^{N_{sc}} \sqrt{W_k x_{n,k}} g_{n,k}(t), $$

(5.4)

$s(t)$ OFDM transmitted signal

$w_k$ power allotment values

$x_{n,k}$ complexed valued data symbol

$g_{n,k}$ waveform of elementary signal pulse

$N_b$ number of time slots

$N_{sc}$ number of sub channels

The parameter $w_k$ is subjected to the condition,

$$ \frac{1}{N_{sc}} \sum_{k=1}^{N_{sc}} w_k = S_o $$

(5.5)

where $S_o$ represents the average power spectral density at the transmitter.
The OFDM channel capacity $C_{\text{OFDM,CSIT}}$, meaning the theoretical maximum of error free data that can be sent with a signal power of $|\alpha_k|^2$ through a communication channel subjected to additive white Gaussian noise of power $N_0$ is expressed as,

$$C_{\text{OFDM,CSIT}} = \max \frac{1}{N_{\text{sc}}} \sum_{k=1}^{N_{\text{sc}}} ld \left( 1 + \frac{\omega_k |\alpha_k|^2}{N_0} \right)$$  \hspace{1cm} (5.6)

The objective is to maximise the capacity $C_{\text{OFDM,CSIT}}$ under the condition (5.5).

This equation is solved by Lagrange’s method. The cost function is defined by (5.7)

$$J(w_1,\ldots,w_{N_{\text{sc}}},\lambda) = \sum_{k=1}^{N_{\text{sc}}} ld \left( 1 + \frac{w_k |\alpha_k|^2}{N_0} \right) - \lambda \left( \sum_{k=1}^{N_{\text{sc}}} \omega_k - N_0 S_o \right)$$ \hspace{1cm} (5.7)

Here, $\lambda$ is the Lagrange’s multiplier. Solving this equation (5.7) yields (5.8)

$$w_k = \max \left\{ \left( \lambda \ln 2 \right)^{-1} - \frac{N_0}{|\alpha_k|^2}, 0 \right\}$$ \hspace{1cm} (5.8)

where, $|\alpha_k|^2$ denotes the gain of $k^{th}$ sub-channel and $N_0$ denotes the noise power. The 1st term $(\lambda \ln 2)^{-1}$ should be carefully chosen, such that condition in (5.5) is satisfied.

Optimising the power by water filling principle may be depicted as filling up a basin with water until certain condition is fulfilled. The shape of a basin corresponds to $N_0 / |\alpha|^2$ and the condition is $\sum_{k=1}^{N_{\text{sc}}} \omega_k - S_o N_{\text{sc}} = 0$. The amount of subchannel signal power corresponds to the depth of water in the basin.

![Figure 5.2 Illustration of water filling principle.[1]](image-url)
Figure 5.1 shows the power coefficients by applying water filling principle for fixed channel realisation and different values of SNR.

It can be seen that,

- For both SNRs the power allotted decreases with decrease in channel power.
- For increasing values of SNR, the variations of $e_k$ decreases.

Figure 5.4 shows the OFDM channel capacities in a Rayleigh fading channel with and without water filling principle.

![Figure 5.3 Water Filling coefficients for 16-QAM and SNR=6dB.](image)

![Figure 5.4 Water Filling coefficients for 16-QAM and SNR=13dB.](image)
(a) without water filling principle to a Rayleigh fading channel.

(b) without water filling principle to a Rayleigh fading channel.

Figure 5.5 Water filling principle – channel capacities [2]
It can be seen from the Figures 5.4 (a) and (b) that

- water filling algorithm results in marginal gain in capacity.
- At higher SNR, the curves coincide

5.1.2. Optimised Power Control Algorithm

Practical systems operate far below the channel capacity with moderate complexity. Hence the adaptation techniques should therefore be based on predetermined coding and modulation methods rather than accomplishing a “water-filling” principle for marginally increasing the theoretical capacity. [1].

The gray encoded signals ensures minimum hamming distance between the neighbouring constellation points. The Figure 5.5 shows the equivalent channel model for BICM-OFDM system with ideal interleaving. System consists of m parallel independent and memoryless binary input/output channels.

Since it is an ideal interleaver & binary symmetric channel with certain transition probability, the exact error probability can be determined at each binary input channel.

![Binary symmetric channel](image)

**Figure 5.6** Binary symmetric channel.[6]
\[ P_{h,\text{QPSK}}^k = Q\left(\sqrt{\gamma_k}\right), \] (5.9)

where, \( Q(x) \) denotes the Q-function.

\[ Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{y^2}{2}\right) dy, \] (5.10)

For adaptive power loading, it is assumed that

- The channel is slow fading, time invariant for the duration of certain OFDM symbols
- SNR of the \( k^{th} \) subcarrier is given by \( \gamma_k = \frac{e_k |\alpha_k|^2}{N_0} \) where \( k = 1,2,\ldots,N_{\text{SC}} \) and \( |\alpha_k|^2 \) represents the channel gain which is perfectly known at the transmitter and at the receiver.

For a given modulation order \( m \), the task of adaptive power loading module is to allot relative signal energy of the \( k^{th} \) subcarrier denoted by \( e_k, k = 1,2,\ldots,N_{\text{SC}} \). The choice of \( e_k \) is subjected to the constraint,

\[ \frac{1}{N_{\text{SC}}} \sum_{k=1}^{N_{\text{SC}}} e_k = S_o, e_k \geq 0, \] (5.11)

Let,

\[ d_e(e) = \frac{1}{N_{\text{SC}}} \sum_{k=1}^{N_{\text{SC}}} e_k - S_o, \] (5.12)

The objective to minimise BER \( P_{h,\text{QPSK}}^k \) under the constraint (5.11) can be solved by Lagrange’s method. The objective function is defined by,

\[ J(y) = P_{h,\text{QPSK}} - \lambda d_e(e) \] (5.13)

where \( \lambda \) is the Lagrange’s multiplier. To minimise \( P_{h,\text{QPSK}}^k \) the gradient of the objective function is equated to zero.

\[ \nabla_y J = 0 \] (5.14)
By solving the above equation,

\[ A_k \exp (e_k B_k) e_k = \Lambda \]  \hspace{1cm} (5.15)

where \( \Lambda \) is a positive constant, \( A_k, B_k \) are factors in the power allocation algorithm, which is given by

\[ A_k = M_k(M_k - 1)/(\sqrt{M_k} - 1)^2 | \alpha_k |^2 \]  \hspace{1cm} (5.16)

\[ B_k = 3 | \alpha_k |^2 / (M_k - 1)N_0 \]  \hspace{1cm} (5.17)

where \( M_k \) is the size of the signal set. Solving (5.15) yields,

\[ e_k = \frac{1}{B_k} W\left( \frac{B_k}{A_k} \right), k = 1, \ldots, N_{sc} \]  \hspace{1cm} (5.18)

where \( W(\bullet) \) denotes a Lambert function and is the inverse of \( f(w) = we^w \) as shown in Figure 5.7.

![Figure 5.7 Lambert’s W function.[2]](image)

The Figure 5.6 can be interpreted as for small values of \( |\alpha_k|^2 \) (less than 1) the \( W(x) = x \) and for a higher range of \( |\alpha_k|^2 \) (greater than 1), \( W(x) \) is flat.

Adaptive power loading algorithm can be interpreted as
Thus it can be seen that channel inversion takes place only after a critical value of $|\alpha_s|^2$. Based on the above equations, the adaptive power loading algorithm can be derived as shown in the Figure 5.7.

\[
e_k \sim \begin{cases} 
|\alpha_s|^2, & \text{for small } \\
(\frac{|\alpha_s|^2}{\alpha_s})^{-1}, & \text{for large }
\end{cases} \quad (5.19)
\]

\textbf{Figure 5.8} Adaptive power loading algorithm.
Figure 5.9 Adaptive power loading for QPSK, 200 Mbps, Eb/No 3 dB.

Figure 5.10 Adaptive power loading for QPSK, 200 Mbps, Eb/No 6 dB.
Figures (5.8) to (5.10) show the behaviour of APL coefficients w.r.t channel spectrum for 3 different $E_b/N_0$ value. Same channel spectrum is considered in all the 3 cases. The graph is plotted for all the data subcarriers(100). It can be seen that the critical value for which channel inversion takes place decreases for higher the $E_b/N_0$ values.

One of the main advantage of power loading scheme is, the receiver does not require any information on the power loading coefficients.

5.2 Adaptive Bit Loading

In adaptive bit loading, the size of the signal set is varied according to the channel conditions resulting in adaptive modulation techniques. Heuristic approach on adaptive bit loading is discussed in [2]. The objective is to minimise the overall BER at the decoder output by optimising the size of signal set. Adaptive bit loading, can be applied to a frequency selective, slowly varying channel.
If the total number of bits per OFDM symbol is given by \( M_s \) and the size of signal set given by \( m_k \), then the choice of \( m_k \) is subjected to the condition

\[
\sum_{k=1}^{N} m_k = M_s \tag{5.20}
\]

There are various methods of adaptive bit loading discussed in [2]. In MBOA-system, there are basically 2 types of constellation mapping – QPSK and Dual Carrier Modulation (DCM). The adaptive bit loading is considered only for QPSK since DCM is one of the unique features of MBOA-system.

With QPSK, the size of the signal set in each subcarrier is 2. For adaptive bit loading,

\[
m_k = \{0, 2, 4\} \tag{5.21}
\]

If the subcarrier power is worse, then no information is transmitted across that subcarrier which means the size of the signal set is 0 (no modulation). This information is loaded to a subcarrier which has highest power. Hence the size of the signal set of this subcarrier becomes 4 (16-QAM). For all other intermediate subcarriers, the size of the signal set remains 2 (QPSK).

Further, adaptive bit loading is combined with adaptive power loading. The adaptive power loading can be modified in such a way that no power is allocated to the subcarrier which carries no information. This offers a higher power to the better subcarriers compared to the previous scenario without adaptive bit loading. The modified optimum power loading algorithm is illustrated in Figure 5.11. The adaptive bit loading algorithm is illustrated in Figure 6.11, chapter 6.
Figure 5.12 Adaptive power loading algorithm when adaptive bit loading is employed.

\[
\Lambda = \Lambda.E_s/E_{s0}
\]

\[
1/Nsc \sum \Lambda/A_k \leq E_s
\]

\[
\text{Compute } \Lambda \text{ which fulfills } 1/Nsc \sum \Lambda/A_k \leq E_s
\]

\[
\text{Calculate the } E_{s0} = 1/Nsc \sum 1/B_k W(\Lambda B_k/A_k)
\]

\[
\text{Is } E_k, E_{s0} \text{ close?}
\]

\[
\text{computed } e_k \text{ values are APL coefficients}
\]

\[
\text{start}
\]

For all subcarriers

If subcarrier has no information

\[
\text{Compute } e_k
\]

No power is allocated, next subcarrier

If subcarrier has no information

Note:
Here Nsc excludes the channel which carries no information
6.0 Implementation and Simulation results

6.1 Introduction to SystemC

SystemC provides hardware-oriented constructs within the context of C++ as a class library implemented in standard C++. It provides an interoperable modelling platform which enables the development and exchange of very fast system-level C++ models. It also provides a stable platform for development of system-level tools [9].

SystemC is a freeware library of C++ classes and a event based scheduler. A SystemC user produces modules which implements functions. These modules are C++ classes. This implies that the simulator developed is object oriented since SystemC is a library of C++ classes. Each module of the simulator is used to host a specific algorithm.

<table>
<thead>
<tr>
<th>Methodology Specific Libraries</th>
<th>Layered Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master/Slave library, etc</td>
<td>Verification library, TLM library, etc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primitive Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal, Fifo, Mutex, Semaphore, etc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Structure Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modules</td>
</tr>
<tr>
<td>Ports</td>
</tr>
<tr>
<td>Interfaces</td>
</tr>
<tr>
<td>Channels</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 valued logic</td>
</tr>
<tr>
<td>Bits and Bit Vectors</td>
</tr>
<tr>
<td>Arbitrary Precision Integers</td>
</tr>
<tr>
<td>Fixed Point types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event Driven Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
</tr>
<tr>
<td>Processes</td>
</tr>
</tbody>
</table>

| C++ Language Standard |

Figure 6.1 SystemC architecture.
The Figure 6.1 shows SystemC core architecture. The gray portion represents the SystemC core language standard. The layers above or top of SystemC consists of design libraries and standards apart from SystemC.

An event driven simulator forms the base of SystemC. It works with events and processes. Ports and modules are used to represent the structure. Interfaces and channels facilitates communication. Library includes a set of built-in primitive channels that have wide use such as signals and FIFO.

SystemC system consists of sets of one or more modules. Each module describes an algorithm using ports, interfaces, channels, internal data. The communication between processes inside different modules are enabled via ports, channels, interfaces. All processes occur concurrently.

Modules use FIFO to pass data from one to another by means of ports. There can any number of ports between 2 modules.

Each Module shall implement only a single class. This class name would be the module name. Each port can be specified with a particular data type.

Each module consists of 5 private methods apart from the main method (Main()).

- **Reset()** – sets the variables in the corresponding module to a invalid value
- **AssignParameters()** – If the module uses the configuration parameters, this function assigns the required configuration parameters to local variables
- **Mem()** – Memory allocation of the input and output arrays of the module. The input and output arrays in most cases store the input and output data which are received/sent via ports
- **Init()** – Initialisation of the algorithm parameters. These parameters are defined in the control structure of the algorithm and are initialised in this method
- **Free()** – At the end of simulation, all memory, open files & other resources of the modules are released
6.2 Description of MBOA Simulator

In this simulator there are 5 atomic modules present. The 4 modules are hosted by the “TestBench” module. TestBench module is a particular module which does not have any ports. The main function of the testbench module is to host all other atomic modules. These atomic modules are connected via FIFOs. When the simulation is done, all these simulator modules are deleted. The 5 atomic modules are

- Source Module
- Transmitter Module
- Channel Module
- Receiver Module
- Results Module

Out of these, transmitter, channel and receiver are hierarchical modules of level 2. In a way, these modules, further initiates the hosted modules.

Figure 6.2 shows the entities of “TestBench” module and the corresponding FIFO connections between ports of individual hierarchical modules.

The functionalities and implementation of these hierarchical modules will be discussed in the next section.

6.2.1 Source Module (source)

This module does not host any other modules. The main function of this module is to generate raw bits to the Transmitter module and Results module.
6.2.2 Transmitter Module (TxH2)

This module is a hierarchical module of level 2. This module further instantiates the other modules which forms the transmitter of the MB-OFDM system. Figure 6.3 shows the entities of the transmitter module.

The description of each module has been already explained in the transceiver architecture in chapter 3.

Note that TX_BIT_# indicates that it is a transmitter port of data type bits and TX_CPX_# stands for transmitter port of data type complex symbol.
6.2.3 Channel Module (ChannelH2)

Channel module is a hierarchical module of level 2. This module further initiates 2 modules. The modules hosted by ChannelH2 modules are multipath SISO module and AWGN module. Figure 6.4 shows the block description of this module. The functions of this module are

- To receive the complex symbols from the Transmitter-GI Insert module.
- The information signal is convoluted with the channel coefficients at the SISO module.
- The convoluted signal is corrupted with AWGN at the AWGN module
- The corrupted signal is then sent to Receiver-GI removal module.

6.2.4 Receiver Module

The receiver module is hosted by ‘TestBench’ module. It is a hierarchical module of level 2 which hosts the other modules. The modules hosted by receiver module and their FIFO connections are shown in Figure 6.5.
In the current simulator, the receiver has the perfect channel state information from the Channel module.

**“ChannelH2” Module**

![Diagram of ChannelH2 module](image)

Figure 6.4 Entities of channel “ChannelH2” module.

**“RxH2” Module**

![Diagram of RxH2 module](image)

Figure 6.5 Entities of receiver “RxH2” module.
6.3 Adaptive Power Loading

The modified transmitter module with the adaptive power loading block and the corresponding port connection is shown in Figure 6.6. Also, in the receiver, the soft demapping module is modified to implement hard decision decoding in addition to the existing soft decision decoding.

![Figure 6.6 Transmitter module with adaptive power loading module.](image)

The frame structure of MBOA system is shown in Figure 6.7. The 6 channel estimation sequences are sent by the framing module. The 24 synchronisation symbols (frame + packet synchronisation) symbols are from the OFDM modulation module.

There are basically 100 channel realisations for each of the 4 channels and 3 sub bands which are stored in an external file. Each channel file contains the following parameters

- Band Number
- The centre frequency of each sub band
- The delay spread for each channel type
- The channel IR coefficients for each path
Adaptive power loading can be applied to a slow fading, time invariant channel. Since the coefficients are uncorrelated, the implementation is carried out in such a way that a 2\textsuperscript{nd} temporary channel is introduced and it initialises both the transmitter and the actual SISO channel. By this way, transmitter, channel and receiver have same channel coefficients during each transmission.

This is illustrated in Figure 6.8. Adaptive power loading algorithm is not carried out for the 24 synchronisation symbols. For each frame, during the 1\textsuperscript{st} 24 symbols, the channel coefficients from temporary channel are sent only to the actual SISO channel module. Only after 24 OFDM symbols of each frame, the channel coefficients are sent to both channel and transmitter - adaptive power loading module. This ensures that both the modules are synchronised and has the same channel coefficients. Also, the adaptive power loading is not done for the 1\textsuperscript{st} 6 channel estimation symbols received from framing module.

Further, the port connections in the “TestBench” module has been modified. The temporary channel and the port connections to transmitter and channel module is shown in Figure 6.9. However, the channel coefficients to receiver are sent from the channel as before.
Figure 6.8 Illustration of temporary channel and channel coefficients flow.

The “RxH2.ChannelEst” module in the receiver is the one which receives the IR-coefficients from the SISO channel (“ChannelH2.SISO”) and sends the channel state information to the “Equalizer” module. Thus perfect channel state information is available at the receiver.
If the input signal is $x(t)$, at the output of “AdaptivePowerLoading” module, the input signal is multiplied by the factor of adaptive power loading coefficients (w coefficients). Since the receiver has the CSI directly from the channel module, the w coefficients are not available at the receiver module. This does not affect the BER results for data rates up till 200 Mbps where the modulation scheme is QPSK. In case of Dual Carrier Modulation (DCM), this changes the decision boundary. Hence it is essential that these power loading coefficients are available at the receiver.

Since adaptive power loading is done only for the payload, the power loading coefficients are sent to the “RxH2.ChannelEstimation” module. At the channel estimation module, the
channel coefficients are multiplied with these power loading coefficients and sent to the receiver. This is illustrated in Figure 6.10.

The power coefficients are of long double data type. Rather than introducing a port between the 2 modules, the coefficients are written to an external file and made available to receiver module.

Figure 6.10 Illustration - sending APL coefficients to receiver.
In the configuration file, the options for adaptive power loading parameters are added. The codes are presented in the next section, followed by the simulation results for various data rates.

6.4 Adaptive Bit Loading

Under normal scenario, adaptive bit loading should be done in the transmitter mapping module and at the receiver adaptive demapping. The modulation schemes on each subcarrier should be informed to the receiver for the demapping purposes. In the current MBOA simulator, the output of the mapping module is number of samples per frame and the framing module divides it to the number of samples per OFDM symbol. It is essential to map the channel coefficients and samples corresponding to each subcarrier to perform adaptive bit loading and identify the signal set size on each subcarrier. MBOA system offers various data rates from 53 Mbps to 480 Mbps with different spreading rates such as 4, 2 and 1. This time and frequency spreading is implemented in the framing module at the transmitter.

Size of the data subcarrier is 100. Since spreading is carried out in the framing module and the size of the output samples at mapping module is not equal to the size of data subcarriers, adaptive bit loading cannot be carried out at the mapping module due to practical implementation difficulties.

It was attempted to implement this adaptive bit loading algorithm in the adaptive power loading module. Hence for adaptive bit loading, the functions in adaptive power loading are modified. However the port connections at the transmitter module remain the same. The algorithm for adaptive bit loading is illustrated in Figure 6.11.

Initially, the idea was to consider two subcarriers, one with maximum subcarrier gain and the other with minimum subcarrier gain. This process is followed by adaptive power loading as illustrated in Figure 5.11 in chapter 5.

The receiver should be informed about the change in modulation schemes in both subcarriers. It is also important to identify, where the module which does the reverse of adaptive bit loading should be included. Notice that the adaptive bit loading is done after framing and before OFDM modulation. It is necessary to perform the reverse of bit loading before de-framing. Since for adaptive bit loading, all the 100 information tones are considered, and
hence the reverse process should be done before de-spreading. For this, the receiver module is modified as shown in the Figure 6.12.

**Figure 6.11** Adaptive bit loading algorithm.
"RxH2" Module

The adaptive bit demapping module, receives the channel coefficients from the temporary channel module and performs the same operation as adaptive power loading to identify the subcarriers with minimum and maximum channel gain. Eventually, the attempt failed because of the following reasons,

- At the adaptive bit demapping, the symbols in both subcarriers have to be de-mapped and these bits are to be modulated again to QPSK
- Notice that equalisation is done after the adaptive bit demapping module. Hence the symbols de-mapped are not ensured to be correct as it does not take into account the channel conditions.
- If equalisation is done only for these 2 subcarriers at the reverse adaptive bit loading module itself, then equalisation and demapping for these 2 subcarriers should not be done at the equalizer and demapping module once again. This means, both modules should be informed about the 2 subcarriers.
- Diversity gain by means of time and frequency spreading would be lost if the demapping is done before combining / dispreading.

Due to the above mentioned reasons, the adaptive bit loading attempt was not successful.
6.5 Simulation Results

(a)

(b)
Figure 6.13 Adaptive Power Loading algorithm for same channel conditions and different Eb/No values.
Figure 6.14 200 Mbps, NLOS channel (CM2), with MRC, BER and PER performances
Figure 6.15 200 Mbps, NLOS channel (CM2), with MMSE, BER and PER performances
Figure 6.16 480 Mbps, NLOS channel (CM2), with MMSE, BER and PER performances
Figure 6.17 53.3 Mbps, NLOS channel (CM2), with MRC, BER and PER performances
Figure 6.18 53.3 Mbps, NLOS channel (CM2), with MMSE, BER and PER performances
Conclusion

The Adaptive power loading algorithm to improve the BER performance is successfully implemented in the WiMedia’s MB-OFDM simulator and led to the following conclusions:

- For data rates employing QPSK modulation, the receiver was not informed about the power loading coefficients. This is one of the advantages of power loading as it eliminates the need to inform the receiver.
- However, since the receiver has the perfect channel state information from the channel, the equaliser is not informed about the power loading coefficients. For data rates employing dual carrier modulation, this information is sent to the receiver as it affects the decision boundary for hard decision decoding.
- Implementation of adaptive bit loading failed and the problems were identified.
- Simulation results for various scenarios are presented. Simulations are presented for a NLOS channel. From the simulation results, it can be inferred that:
  - For data rate 53.3 Mbps, the spreading rate is 4 (time spreading – 2, frequency spreading – 2) to achieve diversity gain in time and frequency. As a result, the gain achieved through adaptive power loading is low (0.5 dB).
  - For data rate 200 Mbps, the spreading rate is 2 (time spreading). In this case, the gain achieved through adaptive power loading with MMSE combiner is 1 dB and with MRC is 2 dB.
  - For data rate 480 Mbps, the spreading rate is 1 with dual carrier modulation. The gain achieved through adaptive power loading is 1 dB.
Appendix A

Source code for temporary channel implementation

/*@ ******************************************************************************  
* ###########################################################################  
* ###########################################################  
*  ## Organization         :  
*  ##                      :  
*  ## Module name          : ref_temp_channel  
*  ## File name            : Sc_ref_temp_channel.cpp  
*  ## Language             : C++  
*  ## Short description    : This module is a temporary multipath channel simulator.  
*  ##                      :  
*  ## Hosted by            : 'TempChannelH2' hierarchical module  
*  ## Input signal         : None  
*  ## Output signal        : Frame of complex channel coefficients sent to the 'ChannelH2.siso' module & 'TxH2.AdaptivePowerLoading' module  
*  ##                      : size: number of subcarriers (frequency-domain)  
*  ## Control parameters   : The control parameters needed by the algorithm are encapsulated in a REF_TEMP_CHANNEL_CTRL structure  
*  ##                      : (see the 'ref_temp_channel_funct_lib.h' header file).  
*  ## Called functions     : From 'Main' method: ref_temp_channel()  
*  ##                      :  
*  ## History              : 05.03.07 Created by ComLab, University of Kassel  
*  ## Detailed description : The transmitter module 'TxH2.AdaptivePowerLoading' requires the channel coefficients to perform the adaptive power loading algorithm. Since the channel coefficients are uncorrelated, this module sends the channel coefficients both to 'ChannelH2.siso' & 'TxH2.AdaptivePowerLoading' modules. This way, both have the same coefficients  
*  ##                      :  
*  ## COPYRIGHT            :  
*  ##                      :  
* ###########################################################################  
* ###########################################################  */

// =====  
// Include the module header file of the module  
// =====

#include "Sc_ref_temp_channel.h"

// Method name : ref_temp_channel  
// Description : constructor definition  
// ===============
ref_temp_channel::ref_temp_channel(sc_module_name Name) : sc_module(Name),
FrameLength("FrameLength"),
ChannelType("ChannelType"),
NbCarriers("NbCarriers"),
ChannelNumber("ChannelNumber"),
ChannelThreshold("ChannelThreshold"),
NbOfdmPerFrame("NbOfdmPerFrame")
{
    // Set the thread method
    SC_THREAD(Main);
}

// ============================================================================
// Method name : end_of_elaboration
// Description : this method is automatically called by the SystemC kernel
// before the simulation starts (ie before the "main" method is executed)
// ============================================================================
void ref_temp_channel::end_of_elaboration()
{
    // Reset actions
    Reset();
    // Get the module parameter values
    AssignParameters();
    // Allocation of memory : input/output arrays and control parameter structure
    Mem();
    // Initialization method
    Init();
}

// ============================================================================
// Method name : ~ref_temp_channel
// Description : destructor definition
// ============================================================================
ref_temp_channel::~ref_temp_channel()
{
    // Call the memory release method
    Free();
}
```c++
void ref_temp_channel::AssignParameters()
{
    ChannelType.get_param_value(&TmpChannelType);
    FrameLength.get_param_value(&TmpFrameLength);
    TmpNbCarriers = NbCarriers.get();
    ChannelNumber.get_param_value(&TmpChannelNumber);
    TmpChannelThreshold = ChannelThreshold.get();  // Channel threshold
    TmpNbOfdmPerFrame = NbOfdmPerFrame.get();
}
```

```c++
void ref_temp_channel::Mem()
{
    // Free previous allocated memory (provided for further design refinements)
    Free();

    // Allocate the memory needed by the control parameters structure
    ptCtrl = new REF_TEMP_CHANNEL_CTRL;
    if ( ptCtrl == 0 )
    {
        cout << "ERROR (ref_temp_channel module): memory allocation failure for time control structure." << endl;
        sc_stop();
    }

    // Memory allocation for the input and output signals considering simulation parameters
    // that have been initialized by the "AssignParameters" method
    ptInSymbols = new COMPLEX[ TmpFrameLength ];
    if ( ptInSymbols == 0 )
    {
        cout << "ERROR (ref_temp_channel module): memory allocation failure for input array." << endl;
        sc_stop();
    }

    ptOutSymbols = new COMPLEX[ TmpFrameLength ];
    if ( ptOutSymbols == 0 )
    {
```
cout << "ERROR (ref_temp_channel module): memory allocation failure for output array." << endl;
sc_stop();
}
}

//===========================================================================
// Method name : free
// Description : memory release method
//               It's mandatory to test if the allocation has been done because
//               this method is called by the "mem" method
//===========================================================================

void ref_temp_channel::Free()
{
    int i;

    // Free the memory allocated for :

    // 1 - the input signal from "Tx" module
    // 2 - the output signal to "Rx" module
    // 3 - the control structures of the module

    if ( ptInSymbols != 0 )
    {
        delete[] ptInSymbols;
    }

    if ( ptOutSymbols != 0 )
    {
        delete[] ptOutSymbols;
    }

    if ( ptCtrl != 0 )
    {
        if ( ptCtrl->ptChannelCoeff != 0 )
        {
            delete[] ptCtrl->ptChannelCoeff;
        }

        if ( ptCtrl->IR_Coeff != NULL )
        {
            for ( i = 0 ; i < NB_MAX_BANDS * NB_CHANNEL_REAL ; i++ )
            {
                if ( ptCtrl->IR_Coeff[i] != NULL )
                {
                    delete[] ptCtrl->IR_Coeff[i];
                }
            }

            if ( ptCtrl->IR_Coeff != NULL )
            {
                delete[] ptCtrl->IR_Coeff;
            }
        }

        if ( ptCtrl->ptDelaySpread != 0 )
        {
            delete[] ptCtrl->ptDelaySpread;
        }
    }
}
void ref_temp_channel::Init()
{
    int i;

    ptCtrl->ptChannelCoeff = new COMPLEX[ TmpNbCarriers ];

    if ( ptCtrl->ptChannelCoeff == NULL )
    {
        cout << "ERROR (ref_siso_channel module): memory allocation failure for channel coefficient array." << endl;
        sc_stop();
    }

    ptCtrl->ptDelaySpread = new int[ NB_MAX_BANDS * NB_CHANNEL_REAL ];

    if ( ptCtrl->ptDelaySpread == NULL )
    {
        cout << "ERROR (ref_siso_channel module): memory allocation failure for delay spread array." << endl;
        sc_stop();
    }

    ptCtrl->IR_Coeff = new COMPLEX*[ NB_MAX_BANDS * NB_CHANNEL_REAL ];

    if ( ptCtrl->IR_Coeff == NULL )
    {
        cout << "ERROR (ref_siso_channel module): memory allocation failure for channel coefficient array." << endl;
        sc_stop();
    }

    for ( i = 0 ; i < NB_MAX_BANDS * NB_CHANNEL_REAL ; i++ )
    {
        ptCtrl->IR_Coeff[i] = new COMPLEX[ MAX_DELAY_SPREAD ];

        if ( ptCtrl->IR_Coeff[i] == NULL )
        {
            cout << "ERROR (ref_siso_channel module): memory allocation failure for channel coefficient array." << endl;
            sc_stop();
        }
    }

    ptCtrl->NbCarriers          = TmpNbCarriers; // used by perfect channel estimation only!
    ptCtrl->ChannelType         = TmpChannelType;
    ptCtrl->ChannelNumber       = TmpChannelNumber;
    ptCtrl->ChannelThreshold    = TmpChannelThreshold;
if ( TmpChannelType != AWGN_CHANNEL )
{
    if ( (FctInitMultipathChannel( ptCtrl ) ) == USER_FAILED )
    {
        /*
        ** The channel has not been set correctly. An error message has
        been displayed.
        */
        cout << "ERROR (ref_siso_channel module): error in
    initialisation function." << endl;
        sc_stop();
    }
}

// ====================================================================
// Method name : reset
// Description : empty method (provided for further design refinements)
// ====================================================================
void ref_temp_channel::Reset()
{
    // Reset the dynamic memory array and structure pointers used by the
    // module
    ptInSymbols  = 0; // input signal from "Channel" input port
    ptOutSymbols = 0; // output signal to "Awgn" module
    ptCtrl       = 0; // control structure of the module in time domain
}

// ====================================================================
// Method name : main
// Description : main module method definition
// ====================================================================
void ref_temp_channel::Main()
{
    while ( true )
    {

        if ( TmpChannelType != AWGN_CHANNEL )
        {
            // Channel convolution
            ref_siso_channel_funct ( ptInSymbols, ptOutSymbols, ptCtrl );

            // Output the channel IR only to "ChannelH2.siso" module
            // through the SystemC communication channel
            // for the 1st 24 OFDM symbols
            if (ptCtrl->OfdmSymbolCounter < 24)
CpxOutPort.write( ptCtrl->ptChannelCoeff , TmpNbCarriers
);

// Output the channel IR  to "ChannelH2.siso" &
"TxH2.AdaptivePowerLoading"
// module through the SystemC communication channel
else
{

  CpxOutPort.write( ptCtrl->ptChannelCoeff , TmpNbCarriers );
  CpxOutPort2.write( ptCtrl->ptChannelCoeff , TmpNbCarriers );
}

/* */
Appendix B

Source code for Adaptive power loading main module

/*
 ******************************************
## Organization         : ComLab, University of Kassel
## Module name          : ref_adaptive_power_loading
## File name            : ref_adaptive_power_loading.cpp
## Language             : C++
## Short description    : This module receives the symbols from "framing" module, channel coefficients from the
                        : temporary channel and performs adaptive power loading algorithm and sends it to the "OFDMmod" module
## Hosted by            : 'TxH2' hierarchical module or 'ModH3'
## Input signal         : 1) Input symbols from 'framing' module
                        : 2) Channel IR coefficients from 'temp_channel' module
## Ouput signal         : One frame of QAM symbols sent to the
                        : 'TxH2.OfdmMod' module.
## Control arguments    : The control information needed by the adaptive power loading is encapsulated into a
                        : REF_ADAPTIVE_POWER_LOADING_CTRL structure (see the 'ref_adaptive_power_loading_funct_lib.h' header file). These parameters are:
                        :       - number of bits per frame per user (NbBitsPerFrame)
                        :       - number of QAM symbols per frame
                        :       - number of bits per QAM symbol (ModulationOrder)
## Called functions     : From 'Main' method:
ref_adaptive_power_loading_funct
## History              : 05.03.07 Created by ComLab-University of Kassel
## Detailed description : Performs adaptive power loading algorithm
## COPYRIGHT
##
 ****************************************** */

// =====
// Include the module header file
// =====

#include "Sc_ref_adaptive_power_loading.h"

// Method name : ref_mapping
// Description : constructor definition
ref_adaptive_power_loading::ref_adaptive_power_loading(sc_module_name Cnt) : sc_module(Cnt), ModulationOrder("ModulationOrder"),
NbCarriers("NbCarriers"), NoiseVariance("NoiseVariance"),
APL_Flag("APL_Flag"),
NbQamPerSymb("NbQamPerSymb"),
NbSymbPerFrame("NbSymbPerFrame"),
NbQamPerSymb("NbQamPerSymb"),
("TimeSpreadingFactor")
{
  // =====
  // Set the thread method
  // =====
  SC_THREAD(Main);
}

ref_adaptive_power_loading::~ref_adaptive_power_loading()
{
  // =====
  // Memory release
  // =====
  Free();
}

void ref_adaptive_power_loading::end_of_elaboration()
{
  // =====
  // Reset actions
  // =====
  Reset();
  // =====
  // Get the module configuration parameters
  // =====
  AssignParameters();
}
// =====
// Allocation of memory: input/output arrays and control parameter
// structure
// =====

Mem();

// =====
// Initialization of algorithm parameters
// =====

Init();

}  // end of 'end_of_elaboration' method

/******************************************************************************
Method name : Reset
Description : method provided for further design refinements
*******************************************************************************/

void ref_adaptive_power_loading::Reset()
{
  // =====
  // Reset the dynamic memory array and structure pointers used by the
  // module
  // Mandatory with Visual C++ 6.0 when a 'Debug' version is produced
  // =====
  ptChannelCoeff = 0;
  ptInSymbols = 0;
  ptOutSymbols = 0;  // output signal sent to the 'Spreading' module
  ptCtrl = 0;
  }  // end of 'Reset' method

/******************************************************************************
Method name : AssignParameters
Description : get the value of the configuration parameters
*******************************************************************************/

void ref_adaptive_power_loading::AssignParameters()
{
  NbCarriers.get_param_value(&TmpNbCarriers);
  ModulationOrder.get_param_value(&TmpModulationOrder);
  TmpNoiseVariance = NoiseVariance.get();
  TmpNbQamPerSymb = NbQamPerSymb.get();  // number of data QAM per OFDM symbols
  TmpAPL_Flag = APL_Flag.get();
}

}  // end of 'AssignParameters' method

/******************************************************************************
Method name : Mem
Description : memory allocation for input/output arrays and control
parameters structure
*******************************************************************************/

void ref_adaptive_power_loading::Mem()
{

// =====
// Free previous allocated memory (provided for further design refinements)
// =====
Free();

// =====
// Memory allocation for the input and output arrays considering simulation parameters
// that have been initialized by the "AssignParameters" method
// =====

ptInSymbols = new COMPLEX[TmpNbCarriers];
if ( ptInSymbols == 0 )
{
    cerr << "\n\nERROR [ref_mapping module]: memory allocation failure for the output array - errno: "
    << errno << "\n\n" << endl;
    sc_stop();
}

ptChannelCoeff = new COMPLEX[TmpNbCarriers];
if ( ptChannelCoeff == 0 )
{
    cerr << "\n\nERROR [ref_APL module]: memory allocation failure for the output array - errno: "
    << errno << "\n\n" << endl;
    sc_stop();
}

ptOutSymbols = new COMPLEX[TmpNbCarriers];
if ( ptOutSymbols == 0 )
{
    cerr << "\n\nERROR [ref_mapping module]: memory allocation failure for the output array - errno: "
    << errno << "\n\n" << endl;
    sc_stop();
}

ptCtrl = new REF_ADAPTIVE_POWER_LOADING_CTRL;
if ( ptCtrl == 0 )
{
    cerr << "\n\nERROR [ref_mapping module]: memory allocation failure for the control structure - errno: "
    << errno << "\n\n" << endl;
    sc_stop();
}

} // end of 'Mem' method

// #################################################################
// Method name: Init
// Description: initializations needed by the implemented algorithm
// #################################################################
void ref_adaptive_power_loading::Init()
{
    // =====
    // Initialization of the control structure used by the algorithm
    // =====
    ptCtrl->ModulationOrder = TmpModulationOrder;
    ptCtrl->NbCarriers = TmpNbCarriers;
    ptCtrl->ptInCoeff = new COMPLEX[TmpNbCarriers];
    ptCtrl->NoiseVariance = TmpNoiseVariance;
    ptCtrl->NbQamPerSymb = TmpNbQamPerSymb; // number of data QAM per OFDM symbols
    TmpNbSymbPerFrame = NbSymbPerFrame.get();
    TmpTimeSpreadingFactor = TimeSpreadingFactor.get(); // MBOA Time Spreading Factor

    if ( ptCtrl->ptInCoeff == NULL )
    {
        cout << "ERROR (ref_APL_channel module): memory allocation failure for channel coefficient array."
             << endl;
        sc_stop();
    }
}

// ####################################################################
// Method name : Free
// Description : memory release method
//               : It's mandatory to test if the allocation has been done
//               : because this method is called by the "Mem" method
// ####################################################################
void ref_adaptive_power_loading::Free()
{
    // =====
    // Free the memory allocated for :
    // - the input/output arrays (input interleaved bits, output QAM symbols)
    // - the control structure of the module
    // =====

    if ( ptInSymbols != 0 )
    {
        delete[] ptInSymbols;
    }

    if ( ptChannelCoeff != 0 )
    {
        delete[] ptChannelCoeff;
    }

    if ( ptOutSymbols != 0 )
    {
        delete[] ptOutSymbols;
    }

    if ( ptCtrl != 0 )
    {
        if ( ptCtrl->ptInCoeff != 0 )
        {
delete[] ptCtrl->ptInCoeff;
}
delete ptCtrl;
}

} // end of 'Free' method

// Method name : Main
// Description : method which supports the thread process
// ######################################################

void ref_adaptive_power_loading::Main()
{
    static int counter = 0;
    int OfdmCtr=-1; // ofdm counter
    while ( true )
    {
        OfdmCtr++;

        if ( OfdmCtr > ( 6 + TmpTimeSpreadingFactor * TmpNbSymbPerFrame ) )
        {
            OfdmCtr = 0;
        }
        // =====
        // Read one frame of channel coefficients from the temporary
        // channel module
        // =====
        CpxInPort2.read(ptCtrl->ptInCoeff, TmpNbCarriers);
        Fft(TmpNbCarriers, FFT_CASE, ptCtrl->ptInCoeff);
        // =====
        // Read one frame of complex symbols from 'Framing' module
        // =====
        CpxInPort.read(ptInSymbols, TmpNbCarriers);
        // =====
        // Adaptive power loading is not done for the 1st 6 CE sequences
        // =====
        if(OfdmCtr < 6)
        {
            for (int i= 0; i<TmpNbCarriers; i++)
            {
                ptOutSymbols[i].I = ptInSymbols[i].I;
                ptOutSymbols[i].Q = ptInSymbols[i].Q;
            }
        }
        else
        {
            // =====
            // Check if Adaptive power loading is chosen in the config. file.
            // =====
            if(TmpAPL_Flag == 0)
            {

            }
        }
    }
}
for (int i = 0; i<TmpNbCarriers; i++)
{
    ptOutSymbols[i].I = ptInSymbols[i].I;
    ptOutSymbols[i].Q = ptInSymbols[i].Q;
}
else
{
    for (int i = 0; i<TmpNbCarriers; i++)
    {
        ptOutSymbols[i].I = ptInSymbols[i].I;
        ptOutSymbols[i].Q = ptInSymbols[i].Q;
    }
    // =====
    // calls the adaptive power loading algorithm function
    // =====
    ref_adaptive_power_loading_funct(ptInSymbols, ptOutSymbols, ptCtrl);
}

// =====
// write one frame of complex symbols to OfdmMod module
// =====
CpxOutPort.write(ptOutSymbols, TmpNbCarriers);

} // end of 'Main' method
Appendix C

Source code of the function implementing the adaptive power loading algorithm

/*
###########################################################################
###############################################
##
## Organization : ComLab, University of Kassel
##
## Function name : ref_adaptive_power_loading_funct_lib
## File name : ref_adaptive_power_loading_funct_lib.c
## Language : C Ansi
## Short description : This function executes the adaptive power loading
algorithm.
##
## Function prototype : void ref_adaptive_power_loading_funct(COMPLEX
*ptTxSymbols, COMPLEX *ptRxSymbols, REF_ADAPTIVE_POWER>Loading_CTRL *ptCtrl
)
## In arguments : int *ptTxSymbols One frame of QAM symbols
## : Size: 'number of subcarriers'
## Out arguments : int *ptRxSymbols One frame of QAM symbols with
weighted coefficients
## : Size: 'number of subcarriers'
## Control arguments : The control information is encapsulated in a
REF_ADAPTIVE_POWER>Loading_CTRL structure (see the
'ref_adaptive_power_loading_funct_lib.h' file)
## : The parameters needed by the mapping algorithm
are:
## : - size of the subcarriers (NbCarriers)
## : - size of the QAM symbols per OFDM symbol
(in complex symbols) (NbQamPerSymb)- not used
## : - modulation order (ModulationOrder)
## : - channel coefficients of
size number of carriers (complex symbols) (ptInChannelCoeff)- not used
## : - noise variance
(NoiseVariance)
## Return parameter : None
## Called from : 'AdaptivePowerLoading' module
## Called functions : long double LambertW (const long double z)
function to compute the lambert W function
##
## History : 05/03/07 ComLab, University of Kassel
## Detailed description : performs adaptive power loading algorithm to
compute the weighted coefficients for
## : each subcarrier based on Channel State
Information (CSI)
##
## COPYRIGHT :
##
########################################################################### */

/* Include the function header file */
/*

#include "ref_adaptive_power_loading_funct_lib.h"

85
void ref_adaptive_power_loading_funct(COMPLEX *ptTxSymbols, COMPLEX *ptRxSymbols, REF_ADAPTIVE_POWER_LOADING_CTRL *ptCtrl) {
    int DataPosition[100] = { 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 84, 85, 86, 87, 88, 89, 90, 91, 92, 94, 95, 96, 97, 98, 99, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 124, 125, 126, 127, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56};
    //int NbCarriers;
    //int ModulationOrder;
    long double An[128], Bn[128], W[128], Alpha[128]; // Alpha[]: Channel Coefficients, W[]: Power assigned to the IB Channel, An, Bn: factors
    long double sumW; // sum of power assigned to all sub-channels
    long double S; // transmit power spectral density
    long double So; // So - Variable converging towards S during Iteration
    long double temp=1; // difference between transmit power and iterative power
    long double lamda = 0.000000001; // positive constant
    long double LambertW (const long double z); // Lambert W function
    long double temp2; // diff between S & So
    int it_counter;
    long double temp3, temp4;
    long double tempChannelCoeff;
    int sizeSignalSet;
    sizeSignalSet = pow(double(2), ptCtrl->ModulationOrder);
    long double S_sum = 0;
    //int ExitValue;
    ofstream APL; // access to the Channel file
    std::ostringstream APLFileName; // used to generate the name of the Channel file
    it_counter=1;
    //ExitValue = USER_FAILED;

    // file created to write the adaptive power loading coefficients for performance evaluation
    APLFileName.str(""");
    #ifdef WIN32
        APLFileName << "/data/APL_data";
    #else
        APLFileName << "/data/APL_data";
    #endif
    APL.open(APLFileName.str().c_str());

    if ( ! APL.is_open() )
    {
        // The opening of the channel file has failed
        cerr << "\n\nERROR [APL function]: cannot open the channel file ";
        sc_stop();
    }

    // computes the sub channel gain
    for (int i=0; i<ptCtrl->NbCarriers; i++)
\begin{verbatim}
    Alpha[i] = sqrt( (ptCtrl->ptInCoeff[i].I * ptCtrl->ptInCoeff[i].I) + (ptCtrl->ptInCoeff[i].Q * ptCtrl->ptInCoeff[i].Q));

    // Adaptive power loading algorithm begins
    while (temp >= 0.0000000001)
    {
        sumW=0;
        for (int i=0; i<ptCtrl->NbCarriers; i++)
        {
            temp2 = sqrt ( double (sizeSignalSet) ) - 1;
            An[i]=sizeSignalSet* (sizeSignalSet- 1) / ( (temp2*temp2 ) * (Alpha[i] * Alpha[i] )); // computes An values
            Bn[i]=3* (Alpha[i] * Alpha[i]) / ((sizeSignalSet-1) * (ptCtrl->NoiseVariance )); // computes Bn values
            temp4=Bn[i] * lamda / An[i];
            temp3 = LambertW (temp4); // to compute the inverse of wexp(w) function
            W[i] = 1 / Bn[i] * temp3; // weighted coefficients
            sumW=sumW + W[i];
        /* Determine the transmit power spectral density based on Modulation Order */
        switch (ptCtrl->ModulationOrder)
        {
            case 2:
                S=1;
                break;
            case 4:
                S= 1/sqrt( double (2));
                break;
            case 16:
                S = 1/sqrt( double (10));
                break;
        }
        // computed transmit power spectral density for the current iteration
        So = sumW / ptCtrl->NbCarriers;

        // difference between the actual and iterative power spectral density
        temp = S - So;

        // compute new lambda value for next iteration
        lamda = lamda * S / So;
        it_counter++;
    }

    // the transmitted symbols are multiplied with the adaptive power loading coefficients
    for (int j=0; j<ptCtrl->NbCarriers; j++)
    {
\end{verbatim}
ptRxSymbols[j].I = ptTxSymbols[j].I * sqrt(W[j]);
ptRxSymbols[j].Q = ptTxSymbols[j].Q * sqrt(W[j]);
APL << Alpha[j]*Alpha[j] << "  " << W[j] << endl;
}
} /* end of 'ref_adaptive_power_loading_funct' function */
Appendix D

Hard decision demapping - QPSK

#include <stdio.h>
#include <iostream>
#include <vector>
#include <complex>
#include "fct_demapping.h"

int ref_soft_demapping_funct (COMPLEX *ptRxSymbols, float *ptRxSoftBits, DEMAPPING_CTRL *ptCtrl)
{
    int result = USER_SUCCESS;
    int n_symbols = ptCtrl->num_symbols;
    int n_bits = ptCtrl->num_bits;
    int modulation_order = ptCtrl->modulation_order;

    for (int i = 0; i < n_symbols; i++)
    {
        float real_part = ptRxSymbols[i].real;
        float imaginary_part = ptRxSymbols[i].imag;

        float magnitude = sqrt(real_part * real_part + imaginary_part * imaginary_part);
        float angle = atan2(imaginary_part, real_part);

        float in_phase_bit = angle / (2 * M_PI);
        float out_of_phase_bit = (1 - in_phase_bit) / 2;

        ptRxSoftBits[i * 2] = in_phase_bit;
        ptRxSoftBits[i * 2 + 1] = out_of_phase_bit;
    }

    return result;
}

// More comments and details about the function implementation...
int ref_soft_demapping_funct(COMPLEX *ptRxSymbols, float *ptRxSoftBits,
REF_DEMAPPING_CTRL *ptCtrl)
{
    int SymbolCounter;      /* counter over the QAM symbols */
    int OfdmCounter;        /* counter over the OFDM symbols */
    int BitCounter;         /* counter over the decoded bits */
    int RetValue;           /* return value of the function */
    int NbBitsPerSymb;
    int NoBit;
    int indexRxN;
    int indexRxNplus50;

    NbBitsPerSymb = ptCtrl->NbQamPerSymb * 2;

    /* ==-------------------------------------------------*/
    /* Initialization of the return value of the function */
    /* ==-------------------------------------------------*/
    RetValue = USER_FAILED;

    /* ===------------------------------------------------*/
    /* Simplified hard decision and soft demapping */
    /* ==------------------------------------------------*/

    BitCounter = 0;
    if ( ptCtrl->ModulationOrder == QPSK_MODULATION )
    {
        // hard decision
        if (ptCtrl->Demapping_Flag==0)
        {
            for ( BitCounter = 0, SymbolCounter = 0 ; SymbolCounter < ptCtrl->NbQamPerFrame ; SymbolCounter += 2 )
            {
                if (tempI > 0 && tempQ > 0)
                {
                    ptRxSoftBits[BitCounter] = 1;
                    ptRxSoftBits[BitCounter + 1] = 1;
                }

                if (tempI > 0 && tempQ < 0)
                {
                    ptRxSoftBits[BitCounter] = 1;
                    ptRxSoftBits[BitCounter + 1] = -1;
                }

                if (tempI < 0 && tempQ < 0)
                {
                    ptRxSoftBits[BitCounter] = -1;
                    ptRxSoftBits[BitCounter + 1] = 1;
                }

                //cout >> " Input Symbols " << ptRxSymbols[SymbolCounter] << endl;
                double tempI, tempQ;
                tempI=0;
                tempQ = 0;
                tempI = ptRxSymbols[SymbolCounter].I;
                tempQ= ptRxSymbols[SymbolCounter].Q;
                if (tempI > 0 && tempQ > 0)
                {
                    ptRxSoftBits[BitCounter] = 1;
                    ptRxSoftBits[BitCounter + 1] = 1;
                }

                if (tempI > 0 && tempQ < 0)
                {
                    ptRxSoftBits[BitCounter] = 1;
                    ptRxSoftBits[BitCounter + 1] = -1;
                }

                if (tempI < 0 && tempQ < 0)
                {
                    ptRxSoftBits[BitCounter] = -1;
                    ptRxSoftBits[BitCounter + 1] = 1;
                }

                //cout >> " Input Symbols " << ptRxSymbols[SymbolCounter] << endl;
                double tempI, tempQ;
                tempI=0;
                tempQ = 0;
                tempI = ptRxSymbols[SymbolCounter].I;
                tempQ= ptRxSymbols[SymbolCounter].Q;
                if (tempI > 0 && tempQ > 0)
                {
                    ptRxSoftBits[BitCounter] = 1;
                    ptRxSoftBits[BitCounter + 1] = 1;
                }

                if (tempI > 0 && tempQ < 0)
                {
                    ptRxSoftBits[BitCounter] = 1;
                    ptRxSoftBits[BitCounter + 1] = -1;
                }

                if (tempI < 0 && tempQ < 0)
                {
                    ptRxSoftBits[BitCounter] = -1;
                    ptRxSoftBits[BitCounter + 1] = 1;
                }
            }
        }
    }
}
ptRxSoftBits[BitCounter + 1] = -1;
//cout << "Output Soft bits" << ptRxSoftBits[BitCounter] << " ,
" << ptRxSoftBits[BitCounter + 1] << endl;
}
if (tempI < 0 & tempQ > 0)
{
ptRxSoftBits[BitCounter] = -1;
ptRxSoftBits[BitCounter + 1] = 1;
//cout << "Output Soft bits" << ptRxSoftBits[BitCounter] << " ,
" << ptRxSoftBits[BitCounter + 1] << endl;
}

// soft decision
if (ptCtrl->Demapping_Flag==1)
{
for ( SymbolCounter = 0 ; SymbolCounter < ptCtrl->NbQamPerFrame ; SymbolCounter++ )
{
ptRxSoftBits[BitCounter] = (float) ( sqrt(Kmod[ptCtrl->ModulationOrder / 2]) * ptRxSymbols[SymbolCounter].I );
ptRxSoftBits[BitCounter + 1] = (float) ( sqrt(Kmod[ptCtrl->ModulationOrder / 2]) * ptRxSymbols[SymbolCounter].Q );

BitCounter += ptCtrl->ModulationOrder;
}
}
if ( BitCounter > ptCtrl->NbBitsPerFrame )
{
fprintf(stderr, "\nERROR [ref_soft_demapping_funct function]: overrun of the output bit array (BitCounter = %d)\n\n", BitCounter);
goto end;
}
} /* end of 'for ( SymbolCounter = 0 ; SymbolCounter < ptCtrl->NbQamPerFrame ; SymbolCounter++ )' loop */

RetValue = USER_SUCCESS;
end:
return RetValue;
Hard decision demapping – Dual Carrier Modulation

/*
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #大人会話

int ref_demapping_DCM(COMPLEX *ptRxSymbols, float *ptRxSoftBits, REF_DEMAPPING_CTRL *ptCtrl)
{
    int SymbolCounter; /* counter over the QAM symbols */
    int BitCounter; /* counter over the decoded bits */
    int RetValue; /* return value of the function */
    int g_k ;
    float Kmod_DCM = sqrt(10) / 5; // Normalisation Factor

    g_k  = 0; // g(k), cf. ECMA-368

}
/* Initialization of the return value of the function */
/* ============================================================== */

RetVal = USER_FAILED;

BitCounter = 0;
SymbolCounter = 0;
int tempRxSymbolI[200], tempRxSymbolQ[200];
// ----------
// Hard decision
// ----------

if(ptCtrl->Demapping_Flag==0)
{
    while (BitCounter < ptCtrl->NbBitsPerFrame)
    {
        for ( int k = 0; k < 50; k++ )
        {
            g_k = ((k<25) ? 2*k : 2*k + 50); // g(k), cf. ECMA-368

            if(Kmod_DCM *(2 * ptRxSymbols[SymbolCounter + k].I + 1 * ptRxSymbols[SymbolCounter + k + 50].I) >= 0)
            {
                if (Kmod_DCM *(2 * ptRxSymbols[SymbolCounter + k].I + 1 * ptRxSymbols[SymbolCounter + k + 50].I)<=2)
                    {ptRxSoftBits[BitCounter + g_k] = 1;}
                else
                    {ptRxSoftBits[BitCounter + g_k] = 3;}
            }

            if(Kmod_DCM *(2 * ptRxSymbols[SymbolCounter + k].Q + 1 * ptRxSymbols[SymbolCounter + k + 50].Q) >= 0)
            {
                if (Kmod_DCM *(2 * ptRxSymbols[SymbolCounter + k].Q + 1 * ptRxSymbols[SymbolCounter + k + 50].Q)<=2)
                    {ptRxSoftBits[BitCounter + g_k + 50] = 1;}
                else
                    {ptRxSoftBits[BitCounter + g_k + 50] = 3;}
            }

            if(Kmod_DCM *(2 * ptRxSymbols[SymbolCounter + k].Q + 1 * ptRxSymbols[SymbolCounter + k + 50].Q) <= 0)
            {

            }
        }
    }
}
if (Kmod_DCM *(2 * ptRxSymbols[SymbolCounter + k].Q + 1 * ptRxSymbols[SymbolCounter + k + 50].Q) >= -2)
{ptRxSoftBits[BitCounter + g_k + 50] = -1;
} else
{ptRxSoftBits[BitCounter + g_k + 50] = -3;
}

if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].I + (-2) * ptRxSymbols[SymbolCounter + k + 50].I) >= 0)
{if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].I + (-2) * ptRxSymbols[SymbolCounter + k + 50].I) <= 2)
{ptRxSoftBits[BitCounter + g_k + 1] = 1;
} else
{ptRxSoftBits[BitCounter + g_k + 1] = 3;
}
}

if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].I + (-2) * ptRxSymbols[SymbolCounter + k + 50].I) <= 0)
{if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].I + (-2) * ptRxSymbols[SymbolCounter + k + 50].I) >= -2)
{ptRxSoftBits[BitCounter + g_k + 1] = -1;
} else
{ptRxSoftBits[BitCounter + g_k + 1] = -3;
}
}

if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].Q + (-2) * ptRxSymbols[SymbolCounter + k + 50].Q) >= 0)
{if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].Q + (-2) * ptRxSymbols[SymbolCounter + k + 50].Q) <= 2)
{ptRxSoftBits[BitCounter + g_k + 51] = 1;
} else
{ptRxSoftBits[BitCounter + g_k + 51] = 3;
}
}

if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].Q + (-2) * ptRxSymbols[SymbolCounter + k + 50].Q) <= 0)
{if (Kmod_DCM *(1 * ptRxSymbols[SymbolCounter + k].Q + (-2) * ptRxSymbols[SymbolCounter + k + 50].Q) >= -2)
{ptRxSoftBits[BitCounter + g_k + 51] = -1;
} else
{ptRxSoftBits[BitCounter + g_k + 51] = -3;
}
}

//cout << ptRxSoftBits[BitCounter + g_k] << " \t" << ptRxSoftBits[BitCounter + g_k + 1] << "\t" << ptRxSoftBits[BitCounter + g_k + 50] << "\t" << ptRxSoftBits[BitCounter + g_k + 51] << endl;

SymbolCounter += 100;
BitCounter += 200;
if(ptCtrl->Demapping_Flag==1)
{
    while (BitCounter < ptCtrl->NbBitsPerFrame)
    {
        for ( int k = 0; k < 50; k++ )
        {
            g_k = ((k<25) ? 2*k : 2*k + 50); // g(k), cf. ECMA-368

            // demapping of DCM
            ptRxSoftBits[BitCounter + g_k] = (Kmod_DCM * (2 * ptRxSymbols[SymbolCounter + k].I + 1 * ptRxSymbols[SymbolCounter + k + 50].I));
            ptRxSoftBits[BitCounter + g_k + 50] = (Kmod_DCM * (2 * ptRxSymbols[SymbolCounter + k].Q + 1 * ptRxSymbols[SymbolCounter + k +50].Q));

            ptRxSoftBits[BitCounter + g_k + 1 ] = (Kmod_DCM * (1 * ptRxSymbols[SymbolCounter + k].I + (-2) * ptRxSymbols[SymbolCounter + k + 50].I));
            ptRxSoftBits[BitCounter + g_k + 51] = (Kmod_DCM * (1 * ptRxSymbols[SymbolCounter + k].Q + (-2) * ptRxSymbols[SymbolCounter + k +50].Q));

            //cout << ptRxSoftBits[BitCounter + g_k] << " \t" << ptRxSoftBits[BitCounter + g_k + 1] << "\t" << ptRxSoftBits[BitCounter + g_k + 50] << "\t" << ptRxSoftBits[BitCounter + g_k + 51] << endl;

        }

        SymbolCounter += 100;
        BitCounter += 200;
    }
}

/* ============= */
/* The algorithm has been executed without errors */
/* ============= */

RetValue = USER_SUCCESS;
end:

return RetValue;
} /* end of 'ref_demapping_DCM' function */
List of Abbreviations

ABL    Adaptive Bit Loading
APL    Adaptive Power Loading
AWGN   Additive White Gaussian Noise
BER    Bit Error Rate
BICM   Bit Interleaved Coded Modulation
CM#    Channel Model (Number)
COFDM  Coded Orthogonal Frequency Division Multiplexing
CSI    Channel State Information
DFT    Discrete Fourier Transform
DCM    Dual Carrier Modulation
DEMUX  De Multiplier
FCC    Federal Communication Commission
FFT    Fast Fourier Transform
FFI    Fixed Frequency Interleaving
FIFO   First In First Out
GI     Guard Index
ICI    Inter Carrier Interference
IDFT   Inverse Discrete Fourier Transform
IFFT   Inverse Fast Fourier Transform
ISI    Inter Symbol Interference
LOS    Line Of Sight
MAC    Medium Access Control
MBOA   Multi Band OFDM Alliance
MB-OFDM Multi Band OFDM
Mbps   Mega Bits Per Second
MCM    Multi Carrier Modulation
NLOS   Non Line Of Sight
OFDM   Orthogonal Frequency Division Multiplexing
PAN    Personal Area Network
PER    Packet Error Rate
PHY    Physical layer
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPSK</td>
<td>Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>SCM</td>
<td>Single Carrier Modulation</td>
</tr>
<tr>
<td>SISO</td>
<td>Single Input Single Output</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>TFC</td>
<td>Time Frequency Codes</td>
</tr>
<tr>
<td>TFI</td>
<td>Time Frequency Interleaving</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>UWB</td>
<td>Ultra Wide Band</td>
</tr>
<tr>
<td>WPAN</td>
<td>Wireless Personal Area Network</td>
</tr>
</tbody>
</table>
Bibliography


[16] Zhongjum Wang, Wenzhen Li, Lee Guek Yeo, Yanxin Yan, Yujing Ting, Masayuki Tomisawa, “A technique for Demapping Dual Carrier Modulated UWB OFDM signals with improved performance”, 0-7803-9152-7/05, 2005 IEEE.