Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Zhonghe SONG

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

Department of Electrical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

High Speed Bling Equalizer VHD2 Desing

T. Yeap

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

A. Karnouch

Q.J. Zitang

J.-M. De Koninck, Ph.D

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

VHDL Design and Implementation of High Speed Blind Equalizer

Zhonghe Song

A Thesis submitted to the Faculty of Graduate Studies and Postdoctoral Studies in partial fulfillment of the requirements for the degree of Master of Applied Science, Electrical Engineering

April 2004

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 0-494-01610-8
Our file Notre référence
ISBN: 0-494-01610-8

Abstract

Blind equalization has significant advantages over conventional adaptive equalization because of its self-recovery ability. This ability makes blind equalization the best choice in broadcasting type communication. With increased demands for broadband data communication, the speed of blind equalization is continuously increasing. The main challenge lies in designing a high speed blind equalizer using current VLSI technology.

Different blind equalization algorithms were investigated for this thesis. Comparisons of their performances and hardware complexities were performed. The CM algorithm offers excellent performance and the most robust property. A pipelined transposed direct form structure equalizer based on the CM blind equalization algorithm was proposed and designed in a VHDL model. Its performance was verified and the finite wordlength effect was investigated. VHDL model was synthesized using 0.25 μm technology.

Acknowledgments

I would like to thank Dr. Tet Yeap, for guiding my research and for reviewing the draft of this thesis.

I also want to thank my wife for her patience and continuous support through the difficulties of these years.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGC	Automatic Gain Control
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CM	Constant Modulus
CMA	Constant Modulus Algorithm
CR	Carrier Recovery
CSA	Carry Save Adder
DC	Design Compiler
DD	Decision Direct
DF	Direct Form
DFE	Decision Feedback Equalizer
DFT	Design For Test
DLMS	Delayed LMS
DPLL	Digital Phase Lock Loop
DRC	Design Rule Check
DVB	Digital Video Broadcasting
FBE	Feed Back Equalizer
FEC	Forward Error Correction
FFE	Feed Forward Equalizer
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSE	Fractional Spaced Equalizer
FSM	Finite State Machine
HDL	Hardware Description Language
HOS	Higher Order Statistics

iid	independent identical distribution
I	In phase
ISI	Inter Symbol Interference
LMS	Least Mean Square
MAC	Multiplier Accumulator
MC	Module Compiler
MSE	Mean Square Error
MMSE	Minimum Mean Square Error
PIPLMS	Pipelined LMS
PRBS	Pseudo Random Bit Sequence
PSK	Phase Shift Keying
PTMP	Point To Multi-Point
Q	Quadrature
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
RLS	Recursive Least Squares
RTL	Register Transfer Level
SNR	Signal to Noise Ratio
SOCS	Second Order Cyclic Statistics
SPW	Signal Processing Workstation
SRRC	Squared Root Raised Cosine
STA	Static Timing Analysis
TDF	Transposed Direct Form
VHDL	VHSIC Hardware Description Language
VLSI	Very Large Scale Integration

1. Introduction

1.1 Motivation

In modern digital communications systems, equalization techniques are widely employed to compensate for the ISI (Inter Symbol Interference) generated by the multipath effect. With the increased demand for network connections, the blind equalization technique, which does not need to transmit a known sequence to train the equalizer at the beginning of the transmission the way a tradition equalizer does, is becoming more and more popular.

In the DVB (Digital Video Broadcasting) system and the PTMP (Point To Multi Point) wireless system, it is not practical to use a conventional method to setup the equalizer. For example, if any one of the multiusers (terminals) lost connection to the central office due to drastic channel changes or simply starts up, the central office would have to resend training signals to let the terminal do the acquisition. This would add overhead to the data transmission or the communication interrupted to all other terminals. For these types of applications, a blind equalizer is the best solution.

The Blind equalization technique is a computation intensive technique. When employed in broadband applications, it is mostly implemented in an ASIC (Application Specific Integrated Circuit) because of its high speed requirement. Utilizing a high speed blind equalizer ASIC design is a challenging task because of the design complexity and speed requirement. The performance and area of the blind equalizer ASIC will be greatly impacted not only by the algorithm and hardware structure but also by the implementation details. An efficient and optimized equalizer ASIC design can only be achieved by controlling algorithm level designs and a detailed implementation. Unfortunately, most research conducted in the past either focused on algorithm levels or on very specific VLSI (Very Large Scale Integration)

features. There was little emphasis on the research of high-speed blind equalizer ASIC designs. This lead to our first motivation: to go through a complete design flow from algorithm selection, RTL coding to logic synthesis in order to design a high-speed blind equalizer ASIC.

During our early research phase, we found that the pipelined transpose direct form structure is a good candidate for the high speed blind equalizer design but there was limited research conducted of the finite word effect on this structure. This factor leads to our second motivation: to study the finite word effect on pipelined transpose direct form high speed blind equalizer.

1.2 Thesis Objectives

The objectives of this thesis are:

1. To design a high speed, low area, high performance blind equalizer ASIC through the blind equalization algorithms selections, hardware structure selections, and VHDL (VHSIC Hardware Description Language) model design and logic synthesis.
2. To investigate the finite wordlength effect on the performance of pipelined transpose direct form blind equalizer structure.

1.3 Summary of Contributions

1. A T spaced and a T/2 spaced high speed blind equalizer based on CM (Constant Modulus) algorithm were implemented and verified using VHDL as design entry. VHDL designs are reusable, synthesizable, and technique independent. As a result, they can be easily applied for different applications with minor modifications.

2. Finite wordlength effects on the pipelined transpose direct form high speed blind equalizer structure were investigated. The results were analyzed and applied to the parameter selection of the equalizer VHDL design.
3. The high speed blind equalizer VHDL designs were synthesized in ST 0.25 μm technology. Timing analysis was performed on the synthesis results, and it showed that 1.5 Gbps data throughput can be achieved by applying this design.

1.4 Research Methodology

The research methodologies applied during the completion of this thesis were algorithm analysis, structure analysis, ASIC RTL (Register Transfer Level) implementation, RTL simulation, and ASIC synthesis. It needs to be indicated that no floor planning, place and route, clock tree insertion or scan insertion were investigated during this research even thought they all belong to the ASIC design flow and largely affect the design performance. The reason for not including them is that those design steps are pure ASIC backend design and have less or no association with equalizer algorithm level designs. Normally, most fabless design companies leave ASIC backend designs to foundry or third party.

VHDL was used as an RTL coding language. We used SPW (Signal Processing Workstation) as the behavior model simulation platform and ModelSim software is used as the RTL model simulation tool. The synthesis tool was Synopsys Design Compiler. The static timing analysis tool was Primetime.

1.5 Thesis Outline

Chapter 1 is the introduction of the thesis.

Chapter 2 presents the background of the research. Different types of blind equalizer algorithms and their structures are presented. Different high speed hardware architectures are given and compared based on their operation speed and hardware complexity. A fractional equalizer structure is reviewed and compared with a T spaced equalizer in terms of its structure, performance, and hardware complicity. A T/2 spaced pipelined transpose form architecture with the CM (Constant Modulus) blind acquisition algorithm is proposed as high speed blind equalizer structure to be further investigated.

Chapter 3 presents VHDL implementation of the proposed blind equalizer. Design details of the proposed equalizer are discussed. The reusable, synthesizable, technique independent RTL design methods based on advanced VHDL features are discussed and applied to the equalizer design. Design details that improve equalizer speed and save chip area are investigated and presented. Finite word effect on the proposed pipelined transpose direct form structure are discussed and verified with RTL simulations. The simulation results also serve as the final parameter selection of the high speed equalizer design.

Chapter 4 presents simulation results and necessary analysis. Behavior model and RTL model simulation results are given in term of BER (Bit Error Rate). Signal constellation plots, which under different channel condition and at different modes (CMA (Constant Modulus Algorithm) lock, CR (Carrier Recovery) lock, DD (Decision Direct) lock), are also presented. Simulations were performed under different channel conditions of AWGN (Additive White Gaussian Noise), multi-path, and frequency offset. Simulation analysis and certain observations are also presented in this section.

Chapter 5 presents the logic synthesis. A synthesis strategy based on advanced Synopsis synthesis tool features is proposed and applied to the VHDL model

developed in this research. Design guidelines are given. Design results in terms of speed and area from traditional synthesis methodology and proposed synthesis methodology are compared to demonstrate the impact of the synthesis techniques on the equalizer performance.

Chapter 6 gives the conclusions and some interesting directions for future research.

2. Background

2.1 Principle of Blind Equalization

Consider a communication system with a time-invariant parameter, which is not necessary but useful to reduce design complexity. Few assumptions are made:

a) The additive noise is Gaussian and white:

b) The information bits are mapped into a complex QAM (Quadrature Amplitude Modulation) symbol x(n) transmitted at time instant n. These symbols are transmitted through a multipath channel, which has impulse response of h(n) and frequency response of H(f). The additive white Gaussian noise v(n) is added to them. At the receiver, AGC (Automatic Gain Control) and a match filter are used for signal detection. The correct symbol clock is recovered by timing recovery circuit. The input symbols of y(n) at the equalizer input then have the form of:

$$y(n) = \sum_{i=-L1}^{L2} h_i \cdot x(n-i) + v(n) \tag{2.1}$$

where $L1$ and $L2$, which are measured in terms of the number of symbols, are the length of the channel pre-cursor and post-cursor sections. The symbols y(n) contain not only the x(n) and z(n), but also contains ISI. The equalizer, as shown in Figure 2.1, in a communication system adaptively adjusts its coefficients to complete the "channel reverting" in order to recover the correct transmitted symbol interrupted by the ISI. The symbols after the equalizer will have the form of:

$$z(n) = \sum_{i=-N}^{N} w_i \cdot y(n-i) \tag{2.2}$$

where, w_i are the equalizer coefficients, and the equalizer has a length of 2N+1.

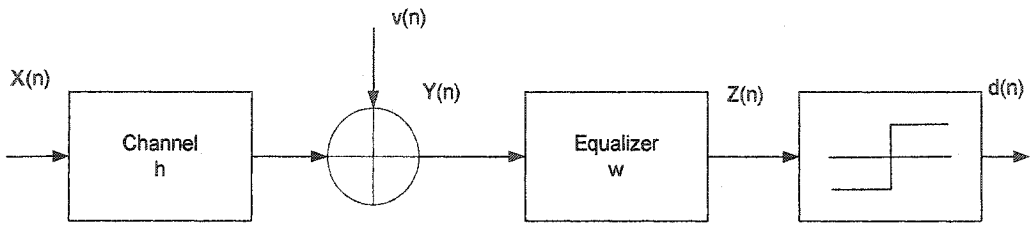

Figure 2.1 Equalized communication systems

Applying equation 2.1, the equation 2.2 becomes:

$$z(n) = \sum_{i=-L1-N}^{L2+N} u_i \cdot x(n-i) + \sum_{i=-N}^{N} w_i \cdot v(n-i) \tag{2.3}$$

$$= u_0 \cdot x(n) + \sum_{i=-L1-N, i\neq 0}^{L2+N} u_i \cdot x(n-i) + \sum_{i=-N}^{N} w_i \cdot v(n-i) \tag{2.4}$$

where $\{u_i\} = \{h_i\} \otimes \{w_i\}$ is the convolution of the channel and equalizer.

If the W(f), the equalizer frequency response, equals to the inverse of H(f), then the U(f), the cascaded system frequency response, will have the form of:

$$U(f) = H(f) \cdot W(f) = H(f) \cdot \frac{1}{H(f)} = 1. \tag{2.5}$$

The cascaded system impulse response $\{u_i\} = \{h_i\} \otimes \{w_i\}$ will have the form of

$$u = (0,\cdots,0,1,0,\cdots 0) \tag{2.6}$$

The ISI, which is the second term in equation 2.4, will become zero and the equation 2.4 will have the form of:

$$z(n) = x(n) + \sum_{i=-N}^{N} w \cdot v(n-i) \tag{2.7}$$

The first term on the right hand side of equation 2.7 is the recovered signal and the second term is the filtered noise.

It is well known that a linear equalizer with MSE (Mean Square Error) criterion has the same property as the Wiener filter [10]. In the Wiener filter theory, to achieve the MMSE (Minimum Mean Square Error), the filter needs to complete the matrix conversion to solve the Wiener-Hopf equation. This matrix conversion is not used in practice. Instead, some iterative search algorithms, such as LMS (Least Mean Square) or RLS (Recursive Least Squares), are used to minimize the cost function since the cost function is a quadratic function of the filter tap weight vector.

Understanding the LMS algorithm will help us understand the blind equalization principle. Let us first examine the LMS algorithm. Figure 2.2 shows its structure.

The output of this equalizer has the form of:

$$z(n) = \sum_{k=0}^{N-1} w^k(n) y(n-k) = w^T y \tag{2.8}$$

$$e(n) = d(n) - z(n), \tag{2.9}$$

where w are the tap coefficients, d is the output of the slicer, and y are the input signals of the equalizer. w and y have the form of:

$$w = [w^0 \; w^1 \; \ldots \; w^{N-1}]^T$$

$$y = [y(n) \; y(n\text{-}1) \ldots \; y(n\text{-}N\text{+}1)]^T$$

y, d and w are complex signals.

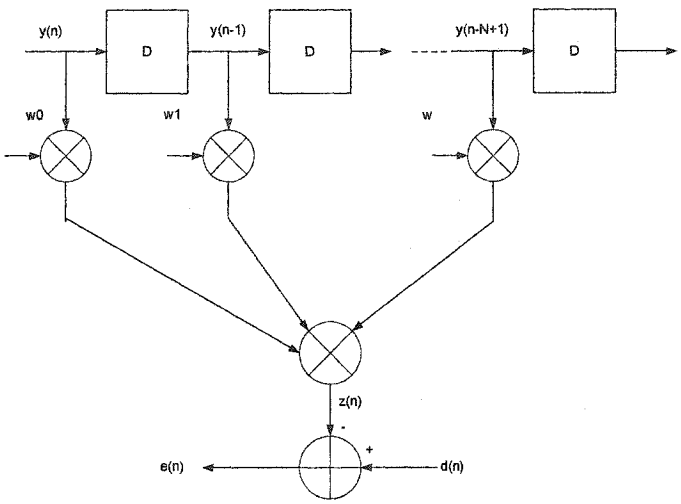

Figure 2.2 Linear equalizer structure

The LMS algorithm is designed to minimize the mean square error and it has the cost function of:

$$J(n) = E([d(n) - z(n)]^2) \tag{2.10}$$

Solving this equation, we get:

$$w(n+1) = w(n) + \mu \cdot e(n) \cdot y^*(n) \tag{2.11}$$

We see that this algorithm updates its weight coefficients by checking its decision error. However, at the very beginning of the coefficient update, we cannot get an accurate estimation (d(n)) of the input signal y(n). As a result, the equalizer cannot function properly. To solve this problem, the traditional approach is to send a known training sequence to train the equalizer. This process is called acquisition mode. The estimated signal approximates the input signal when the equalizer has converged to minimum training error. After the acquisition mode, the equalizer switches to DD mode to use the estimated signal to do the coefficient updating, and the fine tuning, and to compensate the time variation of wireless channels.

In contrast to a conventional equalizer, a blind equalizer does not need a training sequence during the initial acquisition. It uses the input signal's stochastic information to do the coefficient updating.

There are several approaches to do blind equalization. The classical approaches are based on nonconvex optimization. Examples of these are CMA (Constant Modulus Algorithm) [1, 11, 12, 13], Sato algorithm [14], Stop-and-go algorithm [15], and others.

Another approach is based on HOS (Higher Order Statistics) of the input signals explicitly or implicitly [16, 17, 18]. Algorithms of this approach usually optimize some convex functions of HOS to achieve equalization.

A different approach compare with those mentioned above is based on SOCS (Second Order Cyclic Statistics) to be found in over sampled digital communications signals [19, 20, 21]. These SOCS based methods have been shown to have limited practical use due to their restrictive uniqueness conditions and the algorithms' sensitivity to them [22, 23].

Compare with above mentioned approaches, some studies [22, 23, 62] proved that the classical approach is the least complex and reliable in practice. We will direct our focus to these classical approaches and give some detailed description about classical algorithms in next section.

2.2 Algorithm of Blind Equalization

2.2.1 Constant Modulus Algorithm

The constant modulus algorithm was proposed by Godard [11]. Its cost function is defined as:

$$J^{(p)} = E(|z_n|^p - R_p)^2 \tag{2.12}$$

For practical applications, p can be either 1 or 2. The structure will become very simple when p is set to 2. The cost function becomes:

$$J^2 = E(|z_n|^2 - R_2)^2 \tag{2.13}$$

This is similar to the LMS algorithm. Solving this equation, the solution becomes:

$$w(n+1) = w(n) - \mu \cdot e(n) \cdot y^*(n) \tag{2.14}$$

$$e(n) = z(n)(|z(n)|^2 - R_2) \tag{2.15}$$

$$R_2 = \frac{E|x(n)|^4}{E|x(n)|^2} \tag{2.16}$$

From its cost function we can see that the CMA does not compare the difference between z(n) and the estimated input signal d(n) like the LMS algorithm does.

Instead, it compares the $z(n)$ with a constant value R2. Figure 2.3 shows the signal constellation for the QPSK and 16 QAM with the R2.

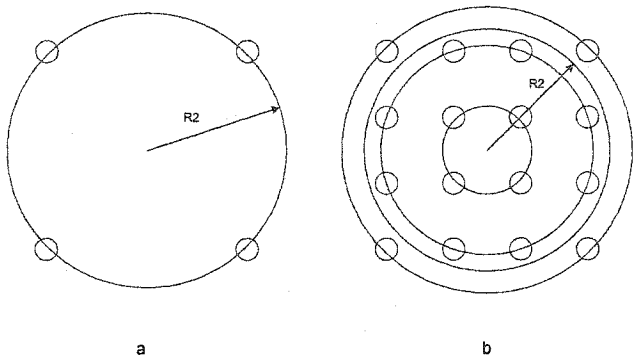

Figure 2.3. (a) QPSK constellation and R2 (b) 16 QAM constellation with R2

The equalizer converges when the equalized signal amplitude matches with its associated mean. The convergence is independent of the signal phase because the CMA constrains only the signal amplitude. The signal phase will be arbitrary even when the equalizer converges. This indicates that the CMA algorithm requires a carrier recovery circuit to correct the equalized signal's phase. This may be the drawback of the CM algorithm. However, due to the fact that the equalizer's convergence is independent of the signal phase, we can put the equalizer in front of the carrier recovery circuit. This configuration has important applications in real communication systems. In a real communication environment, the recovered base band signal unavoidably has frequency offsets and phase offsets due to the finite local oscillator accuracy. The carrier recovery circuit is a necessary block. In a severe multipath channel, before carrier recovery, the recovered signal may become badly distorted and the eye may be completely closed due to the ISI. In this case, the carrier recovery circuit will not work. If we place the equalizer in frond of the carrier recovery, the equalizer will remove most of the ISI and open the eye, because the CMA is independent of signal phase and frequency offset. When the eye is open, the

carrier recovery circuit will be able to correct the frequency offset and phase offset, and the equalizer can be switched from the acquisition mode to the DD mode to do the fine adjustment. This is how CM algorithm is used to do the blind equalization in practice.

For QPSK or other PSK (Phase Shift Keying) signals that have a constant modulus nature, minimizing the CM cost function results in perfect, or zero-forcing equalization. The CM criterion can also be used to successfully equalize the signals whose characteristics are not a constant modulus such as 16 or high order QAM. However, the remaining MSE in these cases will not be very small even when the CMA converges. This can easily be understood through the CMA cost function. Later we will show that perfect equalization can be achieved by using CMA dual mode equalization.

Gardard [11] showed that with the infinite length of equalizer taps and with a small enough step size, the CM algorithm is guaranteed convergence. However the performance curve of the CM algorithm is not convex like a LMS with only one global solution, instead it has many local minima. To force the CMA equalizer to converge to the global minima, we need to let the equalizer start its iteration at a point near the global minimum. Gardard [11] suggested that at the initial state, central taps of the equalizer be set to a high value and all other taps to zero. This method guarantees that the equalizer converges to the global minimum. This is in contrast to the LMS algorithm, which often sets all the taps to zero in the equalizer's initial state.

Ding proved that when the equalizer is not of infinite length, there can be undesirable stable local minima [51]. These minima are length dependent minima. Ding ([28], [29]) showed that a sufficient condition for the existence of global minima is that the equalizer length is at least equal to the channel's delay spread and at the same time, there are no common zeros between subchannels created by considering

the z-transform of the over sampled channel.

There are a number of modified constant modulus algorithms. One of them tries to solve the phase ambiguity problem associated with conventional CMA. Its cost function is:

$$J(n) = E[(|R_e\{z(n)\}|^2 - R_{2,Q})^2 + (|I_m\{z(n)\}|^2 - R_{2,I})^2] \tag{2.17}$$

From this cost function, it comes forth that this algorithm forces both the real part and the imaginary part to a constant value, so that the result phase ambiguity will be 90 degree.

The solution for this algorithm is:

$$w_{n+1} = w_n - \mu \cdot e(n) \cdot y^*(n) \tag{2.18}$$

$$e(n) = R_e[z(n)] \cdot ((R_e[z(n)])^2 - R_{2,R}) + j \cdot I_m[z(n)] \cdot ((I_m[z(n)])^2 - R_{2,I}) \tag{2.19}$$

$$R_{2,R} = \frac{E[(R_e\{x(n)\})^4]}{E[(R_e\{x(n)\})^2]} \tag{2.20}$$

$$R_{2,I} = \frac{E[(I_m\{x(n)\})^4]}{E[(I_m\{x(n)\})^2]} \tag{2.21}$$

The performance of this algorithm is similar to a conventional CMA. Its error calculation function is double in complexity compared to the conventional CMA, and as a result, its hardware structure is more complex than the conventional CMA.

Like a traditional equalizer, the dual mode equalization technique is widely used in the CM algorithm [24]. The dual mode equalization technique uses the CM algorithm as its initial acquisition and when the eye is partially opened, it switches to DD mode to do the channel tracking and remove remaining ISI left by the CM algorithm.

The advantage of the dual mode equalization technique is that the CM algorithm is largely insensitive to small frequency offsets [25]. The equalizer can easily remove most of the ISI caused by multipath and open the eye so that the carrier recovery circuit can correct the phase and frequency offset. After the carrier recovery circuit converges, the equalizer can switch to DD mode to do the fine tuning.

The reason for switching the CMA to DD mode is that after the ISI is significantly reduced, the remaining convergence is much slower than that of DD mode. When switched to DD mode, equalizer can remove remaining ISI much faster [24], and also provide a way to track the time variation of wireless channels. In addition, switching to DD mode results in lower steady state mean square error performance [24], since with few decision errors, the cost function of DD mode is closer to the mean square error cost function than that of CMA [26]. However, the frequency offset correction has to be incorporated in the DD mode [11, 25].

2.2.2 Sato's Algorithm

The Sato's algorithm [14] was originally proposed for PAM signals and has the cost function:

$$J(n) = E[(|z(n)| - \gamma)^2] \tag{2.22}$$

This algorithm also can be extended to QAM and other complex signals. When it is extended to these signals, the cost function becomes:

$$J(n) = E[(|R_e\{z(n)\}| - \gamma)^2] + E[I_m\{z(n)\}| - \gamma)^2] \tag{2.23}$$

The solution becomes:

$$w_{n+1} = w_n - \mu \cdot e(n) \cdot y^*(n) \tag{2.24}$$

$$e(n) = z(n) - \lambda \cdot c\,\mathrm{sgn}(z(n)) \tag{2.25}$$

$$\lambda = \frac{E[R_e\{x(n)^2\}]}{E[R_e\{x(n)\}]} = \frac{E[I_m\{x(n)^2\}]}{E[I_m\{x(n)\}]} \tag{2.26}$$

where:

$$c\,\mathrm{sgn}(z(n)) = syn(R_e\{z(n)\}) + j \cdot syn(I_m\{z(n)\}) \tag{2.27}$$

Compared to the CM algorithm, the Sato's algorithm uses only the sign of the input signal to update its coefficients. From the hardware point of view, the error calculation function of the Sato's algorithm consists only of one subtraction. The CM algorithm needs two multiplications and one subtraction. The Sato's algorithm hardware implementation is simple compared to the CM algorithm. It has a similar performance to the CM algorithm. Its drawback is that there are not only length dependent local minima, like in the CM algorithm, but also algorithm dependent local

minima, which do not exist in the CM algorithm. The Sato's algorithm does not have the frequency insensitive advantage that the CMA has.

2.2.3 Stop-and-Go Algorithm

The Stop-and-go algorithm was proposed by Picchi and Prati in (1987) [15]. This algorithm uses traditional DD algorithm to do the blind equalization. The coefficient update depends on the probability of the decision correctness. If the decision reliability is not high enough, the equalizer will stop the coefficient updating. The stop-and-go algorithm uses the sign of Sato's error value to compare to the sign of decision signal to judge the decision correctness, because there is no training signal to be used as a base to judge the decision correctness. There are two flags, one for the real part and one for the imaginary part, which control the coefficient updating:

$$f_{n,R} = 1 \text{ if } \mathrm{sgn}(R_e[e_{DD}(n)]) = \mathrm{sgn}(R_e[e_{Sato}(n)])$$

$$f_{n,R} = 0 \text{ if } \mathrm{sgn}(R_e[e_{DD}(n)]) \neq \mathrm{sgn}(R_e[e_{Sato}(n)]) \tag{2.28}$$

$$f_{n,I} = 1 \text{ if } \mathrm{sgn}(I_m[e_{DD}(n)]) = \mathrm{sgn}(I_m[e_{Sato}(n)])$$

$$f_{n,I} = 0 \text{ if } \mathrm{sgn}(I_m[e_{DD}(n)]) \neq \mathrm{sgn}(I_m[e_{Sato}(n)]) \tag{2.29}$$

where

$$e_{DD}(n) = z(n) - d(n) \tag{2.30}$$

$$e_{Sato}(n) = z(n) - \lambda \cdot c\,\mathrm{sgn}(z(n)) \tag{2.31}$$

$$c\,\mathrm{sgn}(z(n)) = syn(R_e\{z(n)\}) + j \cdot syn(I_m\{z(n)\}) \tag{2.32}$$

The solution is:

$$w_{n+1} = w_n - \mu \cdot e(n) \cdot y^*(n) \tag{2.33}$$

$$e(n) = f_{n,R} R_e[e_{DD}(n)] + j \cdot f_{n,I} I_m[e_{DD}(n)] \tag{2.34}$$

This algorithm has a similar performance as previously mentioned algorithms, and has a simple hardware structure. Its drawback is slow convergence speed compared to other algorithms. The reason is that this algorithm uses only the sign of the error information to update its coefficient.

2.2.4 Selection of Algorithm

In the selection of blind equalization algorithm, the robustness of the algorithm needs to be considered first. This is because the proposed equalizer will be used in a real environment. The real environment is always changing. If the algorithm selected is not robust, there will be more of a chance that the equalizer will perform badly, or even fail.

The second consideration of the algorithm selection is the performance. The algorithm with good performance will improve overall modem system performance and make other component design simpler (such as use simple FEC design, simple synchronization design) or increase transmission distance (or reduce transmission power).

The third consideration is the potential speed of the algorithm. This is based on the requirement of the application, and needs not to be explained further.

The fourth consideration is the hardware complexity. Hardware complexity is directly associated with ASIC or FPGA chip size. Bigger chip size means higher costs. Bigger design also puts more burdens on ASIC or FPGA synthesis. It may lead the final system design to not meet target timing.

It is understood that the order of the consideration mentioned above could be changed based on the real application requirement.

Based on the analysis of the previous section, the CMA, the Sato's algorithm and the stop-and–go algorithms are suitable blind equalizer algorithms for hardware implementation. Among these algorithms, CM algorithm is the most robust algorithm. It has a simple hardware structure and is suitable for high speed implementation. It draws most of the attention from blind equalization designers. It is widely used in real applications.

From here on, we will focus on the CMA study. Because the CM algorithm is close to the LMS algorithm and the dual mode CMA uses LMS architecture as DD mode. We will build our blind equalizer structure based on LMS architecture. In the next section, the equalizer structure will be discussed with attention to the CMA equalizer structure.

2.3 CMA Blind Equalizer Structure

2.3.1 General structure

The CMA equalizer is normally implemented in a LMS kind of structure, which is shown in Figure 2.2. Some fast convergence structures like Kalman, Fast Kalman and RLS can also be used for CMA algorithm. However these structures have very

high hardware complexity, which is impractical in high speed applications. When the communication speed increases, for the same channel condition, the equalizer length needs to be increased proportionally. Therefore, a truly simple adaptive equalization algorithm like CMA is the best choice.

A CMA blind equalizer is a linear equalizer when in DD mode. The linear equalizer structure is simpler than DFE (Decision Feedback Equalizer) and can easily be implemented as a pipelined structure. However, this kind of equalizer has two drawbacks: noise enhancement and poor performance if the channel condition is severe. One such channel condition is created when there is a deep null in the frequency spectrum. These two drawbacks of the linear equalizer structure can be overcome by using DFE as DD mode equalizer.

The DFE has much better performance compared with linear equalizer [1, 8, 48, 49]. It consists of a FFE (Feed Forward Equalizer) and a FBE (Feed Back Equalizer). The DFE can better handle severe channel condition, without enhancing noise compared with the linear equalizer. The drawback of the DFE is its decision error propagation problem.

At first glance, the DFE is the best choice because of its superior performance. However, due to its nonlinear feature, it is difficult to build the DFE as pipeline structures, which is required when design a high speed equalizer. Therefore, we will only consider the linear equalizer structure in this thesis.

Figure 2.4 is the block diagram of a blind equalizer based on the CMA [11] blind equalization principle. The equalizer consists of 3 main blocks:

1. FIR filter, which is used to perform real channel compensation.
2. Error calculation unit, which consist of two blocks. One block is used to calculate error signal using CMA algorithm in acquisition mode. Another

block is used to calculate the error between the filtered data and the decisional data in DD mode.

3. Tap update unit, which is used to calculate and update the equalizer taps based on the error and the selected algorithm.

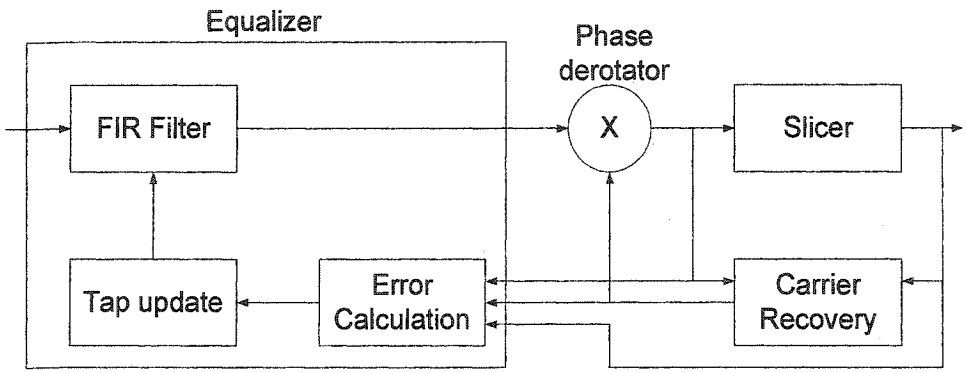

Figure 2.4 Architecture of blind equalizer

Equations 2.8 to equation 2.11 show the signal filtering, error calculation, and tap updating. All 3 operations have to be finished in one clock cycle. The equalizer speed depends on the time spent on completing these 3 operations. We will discuss the high speed blind equalizer structure by their sub blocks.

2.3.2 Traditional LMS Equalizer Structure

There is basically two ways to implement a FIR filter for an equalizer: DF (Direct Form, also known as transversal form) and TDF (Transposed Direct Form) [7], as shown in Figure 2.5 and Figure 2.6.

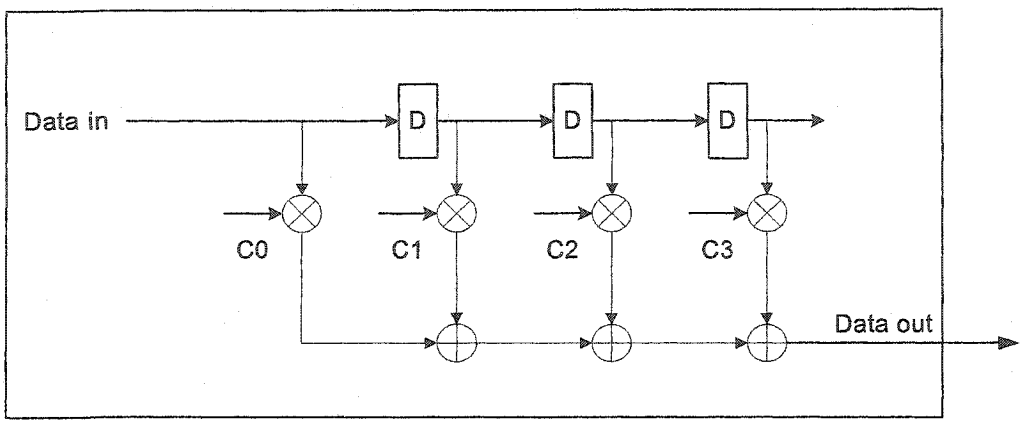

Figure 2.5 Transversal Direct form FIR

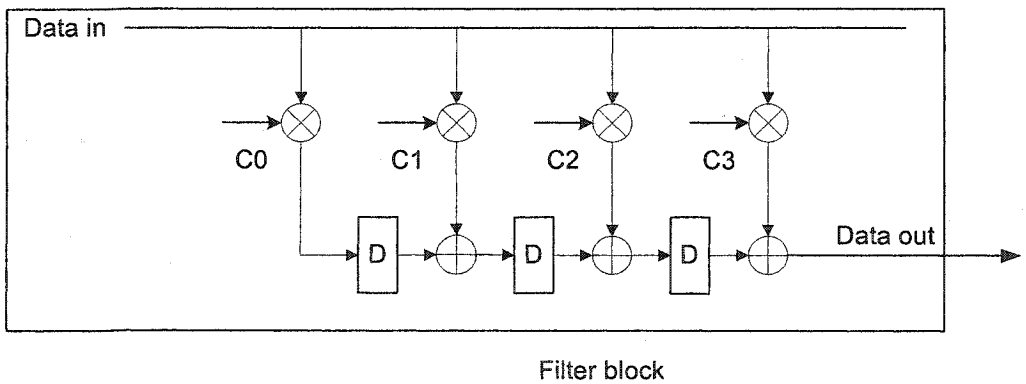

Figure 2.6 Transpose Direct Form FIR

The filter with LMS adaptive algorithm is normally implemented in a transversal form. It has a computation delay or critical path corresponding to the time required to achieve all the multiplication and addition operations. This delay is proportioned to the number of taps. Therefore, a high speed equalizer cannot be realized by using this structure directly.

The TDF, as shown in the Figure 2.6, is a different FIR (Finite Impulse Response) filter architecture compared to DF structure. Using this kind of FIR structure as a filter block for the equalizer may have some advantages over those of direct form FIR filter. The critical path in this filter consists of one multiplication and one addition. With fixed coefficients, this structure produces exactly the same output as a direct form structure. With time-varying or adaptive coefficients, however this structure is different from the direct form structure, as used by the LMS algorithm. Douglas [52] indicated that with a small enough step size, this form works, and its performance is close to standard LMS.

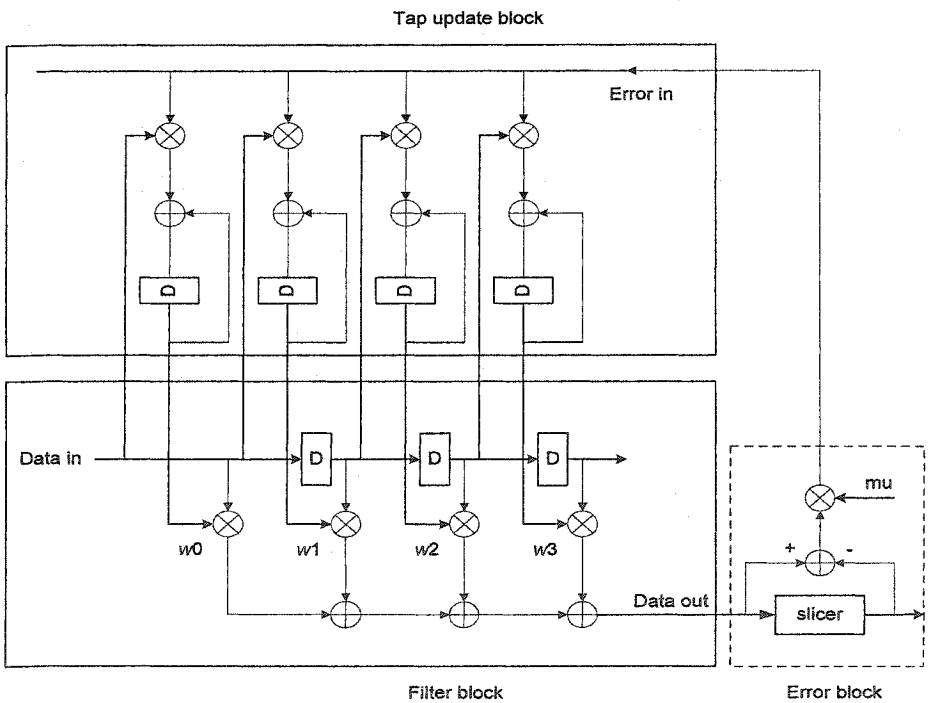

Figure 2.7 Transversal structure equalizer

Figure 2.8 gives a top level view of the FIR filter in an equalizer. Except for a global broadcast input, the transposed form structure is systolic. The equalizer output of transposed form can be described as:

$$Z(n) = \sum_{k=0}^{N-1} W^k(n-k)Y(n-k) \tag{2.35}$$

which is different from the transversal type equalization equation

$$Z(n) = \sum_{k=0}^{N-1} W^k(n)Y(n-k) \tag{2.36}$$

Compared to the standard transversal structure as shown in Figure 2.7, it can be see that these two structures have different operations.

In Figure 2.7, the coefficients $w0$ to $w4$ contribute to the output signal at the same time. In Figure 2.8, $w0$ is not delayed, $w1$ is delayed by one clock cycle, $w2$ is delayed by two clock cycles and $w3$ is delayed by three clock cycles.

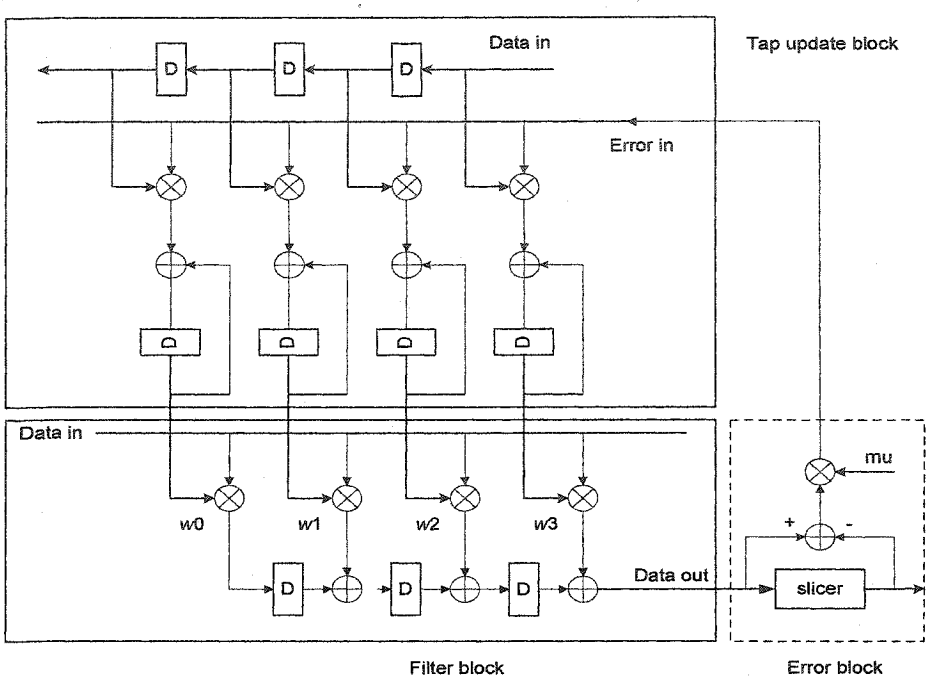

Figure 2.8 Transposed form linear equalizer structure

2.3.3 High Speed LMS Equalizer Structures

There are roughly 3 types of high speed equalizer structures that can be used for the CMA blind equalizer design. One is using TDF FIR structure with the use of pipelined components (multiplier and adder). When building this way, the equalizer speed is decided by the clock cycle that one pipeline stage needs.

The second type is implemented based on the parallel TDF structure [36]. The principle of this structure is similar to TDF but it uses two identical filters to do parallel filtering to double the computation speed. The price to pay for the increase in speed is doubled hardware complexity.

The third structure is implemented by using DF FIR structure with the use of pipelined components (multiplier and adder). This kind of structure is called PIPLMS (Pipelined LMS) [9] or DLMS (Delayed LMS) [41, 42, 43, 44]. This kind of architecture transforms the adders in the filter to their pipelined version. As a result, the error calculation has delays that are equal to the total pipeline stages. The input signal has to be delayed in the same stage to perform the tap updating. There are three types of DLMS:

1. Updating coefficient at the rate of 1/N (N is the number of pipelined delay). This operation is the same as the standard LMS with the same performance as the standard LMS algorithm, except its update speed is reduced to 1/N of the standard LMS algorithm.

2. Updating coefficient at the normal rate but adding some hardware overhead. This architecture has a similar performance as the standard LMS algorithm when small update size is used [9, 41, 42, 43, 44]. The hardware overhead, based on the relaxed look-ahead method [9], is N*(LA -1). N is the tap length of equalizer and LA is the number greater than 1 and less than D. D is the total number of pipelined delay. This algorithm gives a trade off between performance and hardware overhead.

3. Updating coefficient at the normal rate without any hardware overhead. This algorithm has the worst performance compared to other two above. However, its convergence speed is faster than the first one and its hardware is simpler than the second.

The DLMS structure was studied thoroughly. There have been a few researches on pipelined TDF structure. Comparing all the different FIR structures, the pipelined TDF implementation has the simplest hardware structure while achieving high speed.

Parallel TDF will double the computation speed of the basic TDF but requires the doubling of the hardware complexity. However, when pipelined components (multiplier and adder) are used, the speed of the TDF equalizer will be close to the parallel TDF structure and have less hardware complexity. This study will focus on the TDF structure while using pipelined components.

2.3.4 Fractional Spaced Equalizer Structure

Equalizer structures can be classified as either T-spaced or fractional spaced depending on the tap space. All discussions presented so far were only concerned with the T-spaced equalizer.

Compared to T-spaced equalizer, the FSE (Fractional Spaced Equalizer) can effectively compensate for more severe channel distortions with less noise enhancement than the T-spaced equalizer. It is also insensitive to the choice of sampling phase [1, 8, 45, 46, 47], thus lowering the requirement on the timing recovery circuitry. Generally, the FSE performs better than the T-spaced equalizer even when the same number of taps (half the time span) is used.

One explanation of why the FSE has superior performance is that even though the signal arrived at the receiver at symbol rate, the actual bandwidth of the signal is somewhat large. For example, in the case of using Raised Cosine FIR filter as transmit and receive filter with the roll off factor of 0.35, the actual bandwidth is:

$$(1 + 0.35)f = 1.35f.$$

The signal sampled at symbol rate is not enough to satisfy the Nyquist condition. Therefore the T spaced equalizer cannot perform as an optimal filter. The FSE, by its sampling rate, can be synthesized as a combination of a matched filter and a T spaced equalizer. This combination is the optimal receiver structure.

In the study of fractional blind equalization, there is a useful subspace method [29, 30, 50], which indicates that with fractional equalizers, the channel can be decomposed as a multichannel model. For a T/2 spaced equalizer, the channel model can be decomposed containing two subchannels as shown in Figure 2.9, with one even channel and one odd channel respective of the T/2 spaced equalizer decimated to two sub filters.

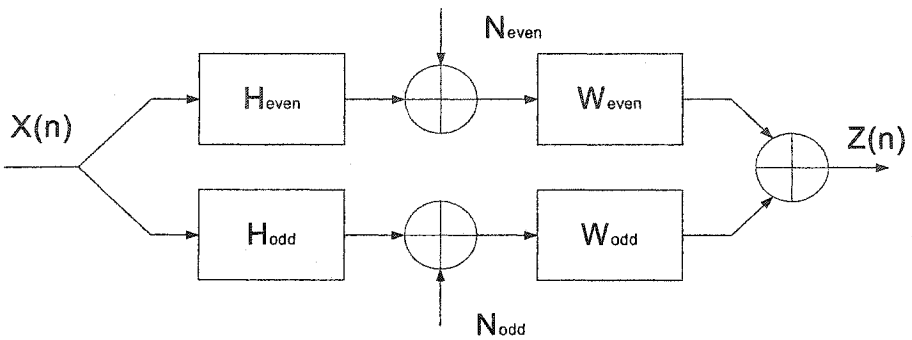

Figure 2.9 Multichannel model

We can implement a zero-forcing equalizer with the following equation:

$$h_{even}(n) * w_{even}(n) + h_{odd}(n) * w_{odd}(n) = \delta(n - v) \tag{2.37}$$

Researches [29, 30, 50] indicate that the solution is valid under the conditions:

1. Two subchannels have no common roots and
2. The decimated equalizer order is equal to the decimated channel order.

This solution implies that if the two subchannels have no common roots (this criteria is easily to be met) and the T/2 fractional equalizer has a tap length that is equal to two times the channel length, the channel will be perfectly equalized. This conclusion will have important implications since we know that the T spaced equalizer will need an infinite tap length to perform the zero-forcing equalization.

From above discussion, we can conclude that a T/2 spaced equalizer has superior performance over T spaced equalizer. In the following study, we will focus on the T/2 spaced equalizer.

2.3.5 Proposed High Speed CMA Blind Equalizer Structure

Based on the discussion so far, we see that the CM algorithm is the most robust, simple and excellent performance algorithm among all blind equalization algorithms. We also see that the pipelined transpose direct form structure can achieve high speeds in addition to having less hardware complexity. We also see that the fractional spaced equalizer has superior performance over the T spaced equalizer. Thereafter, we propose a pipelined transpose direct form structure with CM algorithm as our high speed blind equalizer structure.

2.4 Summary

Blind equalization algorithms have three different approaches: the classical approach, HOS approach, and SOCS approach. From the three approaches, the classical approach has the lowest hardware complexity and is reliable in practice.

The classical approach is based on nonconvex optimization and has three main algorithms: the CMA algorithm, the Sato's algorithm, and the stop-and-go algorithm. They have similar performances but CMA is the most robust and simple algorithm. It also has the advantage that is largely insensitive to small frequency offsets. The Sato's algorithm has the lowest hardware complexity, but is not as robust as CMA and does not have the frequency insensitive advantage the CMA has. The stop-and-go algorithm is very simple but its convergence speed is very low compared to other two algorithms. Compare with other algorithms, the CMA is the most attractive algorithm for practice and will be the target of further investigation.

Dual mode operation of CMA can greatly merit its drawback of slow converge speed, and keep all its advantages.

Blind equalizer hardware structure is normally based on LMS structure. It is mostly implemented as a linear equalizer.

There are 3 types of high speed equalizer structures: DLMS, Pipelined TDF and parallel TDF. Out of these 3 structures, the pipelined TDF has the advantages of high speed and simple hardware structure over other 2 structures. However, the publishing on the pipelined TDF blind equalizer hardware design is difficult to be found. We will focus on this structure in this study.

T spaced and fractional spaced linear equalizers are two different types of linear equalizers. The fractional spaced linear equalizer has a superior performance than the T spaced equalizer in terms of its insensitivity to the timing jitter, and can effectively compensate for more severe channel distortion. The fractional linear equalizer shall be selected as the base structure of the high speed blind equalizer.

Based on the above discussion, a pipelined transpose direct form structure with CM algorithm was proposed as the high speed blind equalizer structure to be further investigated.

3. VHDL Implementation

3.1 Introduction of ASIC Design

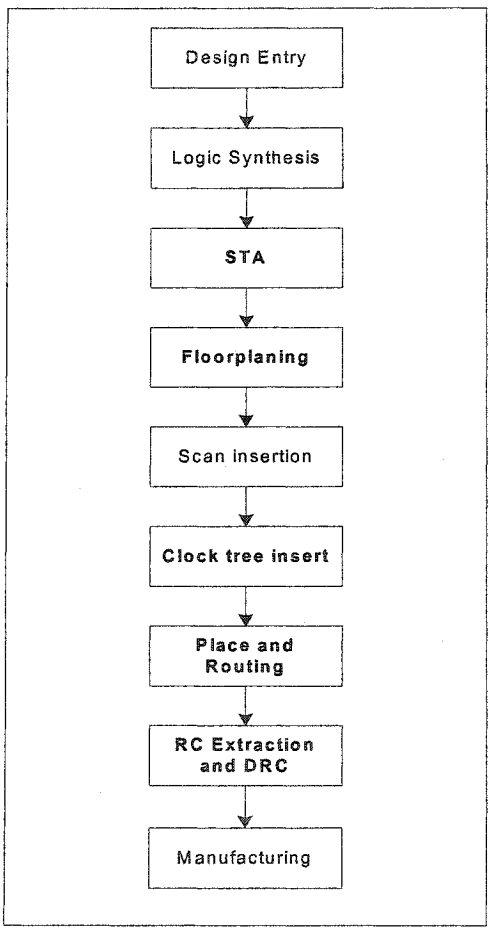

Figure 3.1 ASIC design flow

ASIC design is a technique that uses a HDL (Hardware Description Language), which can be VHDL or Verilog, to describe the hardware structure, and then uses a synthesis tool to do the RTL to gate level conversion. After the synthesis, one can use

backend design tools to do the clock tree insertion, scan insertion, floor planning, place and route. The foundries make a chip base on the final layout information. Refer to Figure 3.1 for a flow chat of an ASIC design. This design technique has the advantages of fast design cycle and cheaper design cost compared to full custom designs. In complex designs having more then 1 million gates, ASIC design is the best choice.

However, compared to full custom design, ASIC chips may run few times slower [55, 56] if they are not designed properly. In this section, we will first derive our detailed equalizer structure and then we will discuss different RTL coding techniques to improve the ASIC design timing.

3.2 Top Level View of Blind Equalizer Hardware Structure

A T spaced blind equalizer block diagram is drown in Figure 3.2 and is based on our proposed high speed blind equalizer structure discussed in previous chapters. The only difference between this figure and Figure 2.8, which is the traditional linear equalizer structure, is in the error calculation block that will be discussed in detail in the following block design section.

3.2.1 T/2 Spaced Equalizer Structure with Hardware Sharing

The fraction spaced equalizer has a similar structure as T spaced equalizer. However, due to the factor that the equalizer output needs to be sampled only at the data sample rate, a tap sharing method can be applied to a fractional spaced equalizer structure to reduce hardware complexity. When tap sharing is applied, the structure shown in Figure 3.3 is derived.

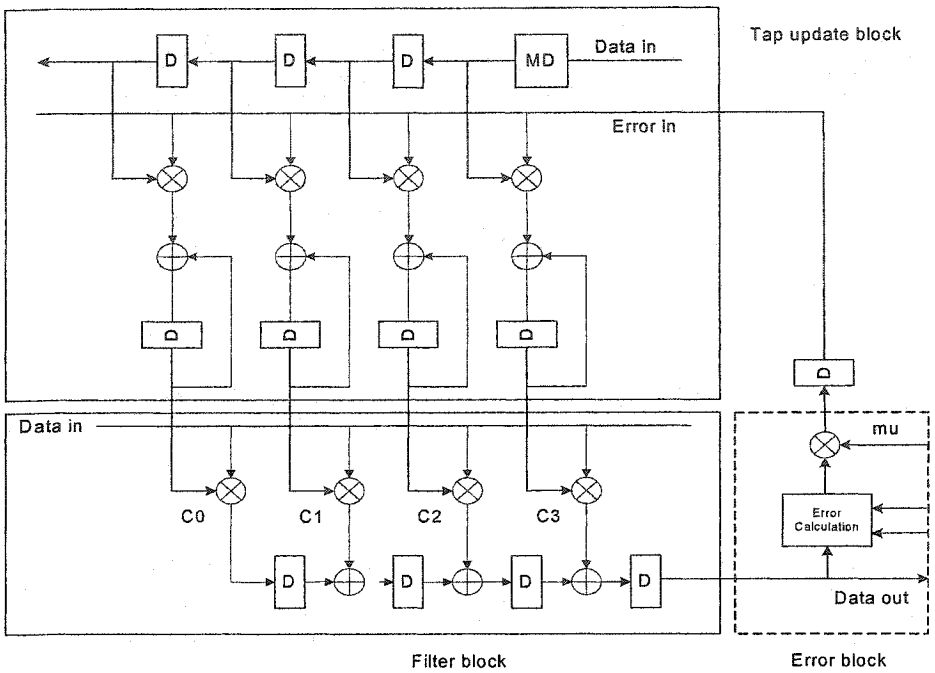

Figure 3.2 Structure of T spaced blind equalizer

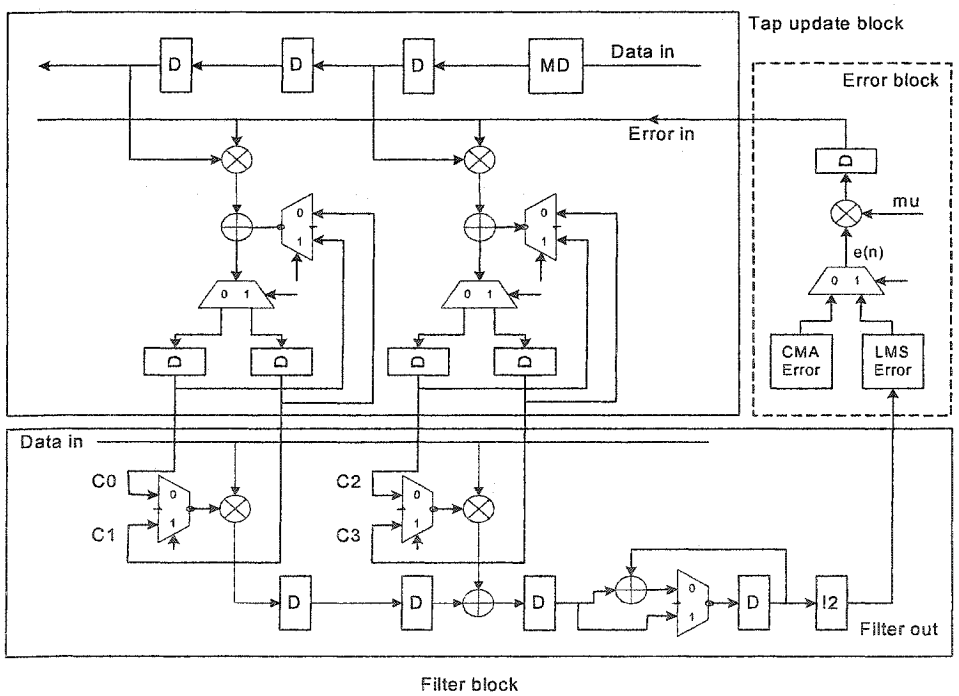

Figure 3.3 Diagram of T/2 spaced equalizer

The tap sharing operation is as follows: the filter runs at sample clock, which is twice the speed of the symbol clock in the T/2 spaced case, and its output is down sampled by 2, and to be sent to the slicer. We can use one sample clock cycle to filter the input signal with odd coefficients and use another clock cycle to filter the input signal with even coefficients. These two results are added by an accumulator and are feet to the slicer. The tap update block can also update the odd and even coefficients in different clock cycles. By applying the tap sharing method, we can save half of the filter multipliers and adders. We can see from Figure 3.2 and Figure 3.3 that our high speed blind equalizer consists of 3 main blocks: the filter block, the tap update block and the error calculation block. We will discuss those blocks in more detail in next section.

3.3 Detailed Block Level Design

In this section, we will discuss about the sub blocks in more detail, and discuss the design of the control block, which, though not shown in this diagram, is necessary for a real hardware equalizer design. It needs to be indicated that the discussions in the following sections focus on fractional equalizer structures. The detailed T spaced equalizer design is derived from the discussion since the T spaced equalizer design is using only subsets of the fractional equalizer design.

3.3.1 Filter Block Design

The transpose FIR filter structure is shown in Figure 2.6. It works with real signals. To make it work with QAM signals, which are complex type, we need to derive a complex filter structure.

The filter operation is based on the equation 3.1:

$$Z(n) = \sum_{k=0}^{N-1} W^k(n-k)Y(n-k) \tag{3.1}$$

where

$$k = 0,1,2,...N-1$$

is the coefficients index of the equalizer and Z(n), W(n) and Y(n) are all complex signals.

This equation can be expanded to:

$$Z(n) = \sum_{k=0}^{N-1} W^k(n-k)Y(n-k)$$

$$= \sum_{k=0}^{N-1} (w_i{}^k(n-k) + jw_j{}^k(n-k))(y_i(n-k) + jy_j(n-k)) \tag{3.2}$$

We denote the footnote of i and j as real and imaginary part respectively.

The left side of equation 3.2 has the form of:

$$Z(n) = z_i(n) + j \cdot z_j(n) \tag{3.3}$$

The right side will become:

$$\begin{aligned} &\sum_{k=0}^{N-1} \{(w_i{}^k(n-k))(y_i(n-k)) - (w_j{}^k(n-k))(y_j(n-k))\} \\ &+ j \cdot \{(w_j{}^k(n-k))(y_i(n-k)) + (w_i{}^k(n-k))(y_i(n-k))\} \end{aligned} \tag{3.4}$$

$$\sum_{k=0}^{N-1} w_i^k(n-k)y_i(n-k) - \sum_{k=0}^{N-1} w_j^k(n-k)y_j(n-k)$$

or

$$+ j \cdot \sum_{k=0}^{N-1} w_j^k(n-k)y_i(n-k) + j \cdot \sum_{k=0}^{N-1} w_i^k(n-k)y_j(n-k) \qquad (3.5)$$

Two different structures can be implemented based on above two equations.

Equation 3.4 forms a single complex FIR filter. Each of the multiplier and adder pairs in this complex filter has the complex form, as shown in Figure 3.4. *Yi*(n) and *Yj*(n) represent the input signals. *Wi*(n) and *Wj*(n) represent the complex taps. *Qi*(n-1) and *Qj*(n-1) represent previous accumulators. *Qi*(n) and *Qj*(n) represent the accumulator output. We can see that one complex multiplier is equivalent to 4 real multipliers and 2 adders/subtracters. One complex adder is equivalent to 2 real adders.

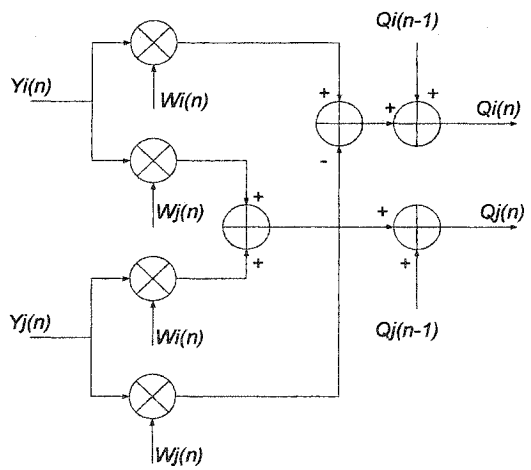

Figure 3.4 The structure of complex multiplier and accumulator pair in the filter

Equation 3.5 forms a structure that consists of 4 real FIR filters. This structure is shown in Figure 3.5.

The real part of the input signal goes though two FIR filters, with the real and

imaginary parts of the taps applied to each of them. The imaginary part of the input signal goes though two FIR filters the same way. The output of these 4 filter are combined to form the complex output signals *Zi*(n) and *Zj*(n).

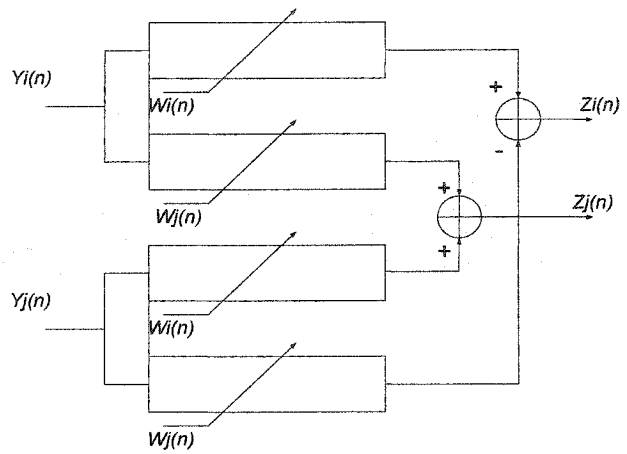

Figure 3.5 The 4 real filters structure

The above two structures have the same function but have different hardware structures. When comparing these two structures, we favor the first one because its tap number can easily be expanded and the timing is easier to meet in ASIC design. Therefore, the given example design will be based on the first structure.

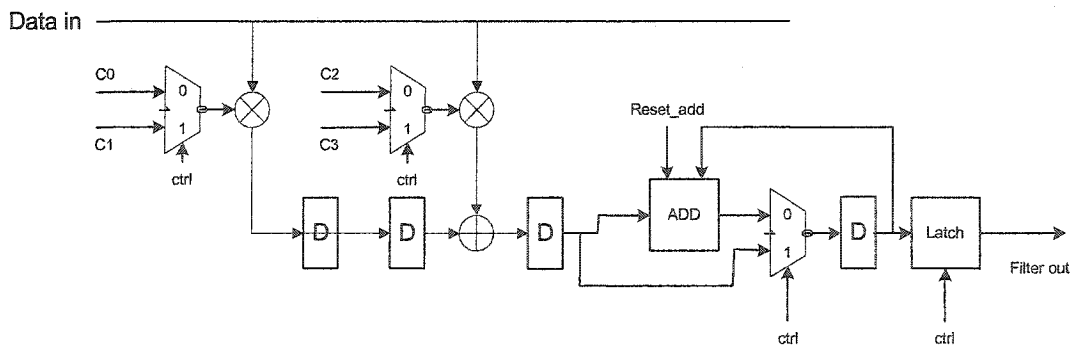

Figure 3.6 Filter block diagram

Figure 3.6 gives the block diagram of the filter block. Only 2 taps are shown in this diagram but they operate as 4 types by sharing the hardware. An adder is included after the filter output to complete the accumulation. This adder accumulates 2 continuous filter outputs and reset every other clock cycle. Its output is latched because this signal will be processed by the error calculation block using half of the clock frequency that the filter block used. The 'ctrl' is the control signal from the control block.

3.3.2 Error Calculation Unit Design

In CMA mode, the error is calculated as:

$$e_{CMA}(n) = z(n)(R^2 - |z(n)|^2) \tag{3.6}$$

In DD mode, the error is calculated as:

$$e_{DD}(n) = (q(n) - a(n))e^{j\hat{\theta}(n)} \tag{3.7}$$

where,

$e_{DD}(n)$ is the complex error signal,

$q(n)$ is the phase derotator output, where $q(n) = z(n) \cdot e^{-j\theta(n)}$,

$a(n)$ is the complex slicer output, and

$e^{j\hat{\theta}(n)}$ is the complex conjugate of the angular correction applied at the phase derotator input.

The diagram of the error calculation in CMA and DD modes are depicted in Figure 3.7 and Figure 3.8 respectively. They both work at the symbol rate since the slicer output update at symbol rate in the fractional equalizer.

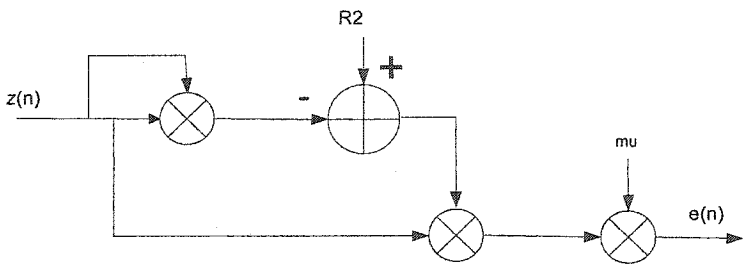

Figure 3.7 Error calculation in CMA mode

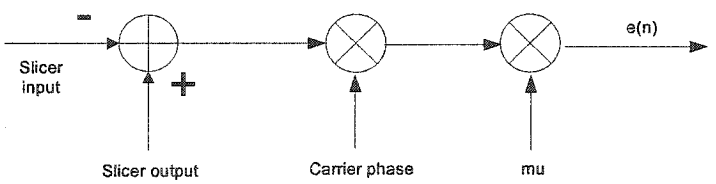

Figure 3.8 Error calculation in DD mode

In these 2 diagrams, the final error outputs are the product of error signals, which are generated by the 2 mentioned equations, multiplied by a 'mu'. 'mu' is the step size and normally appears at the equation of tap update. We move this multiplication from the tap update block to error calculation block because it will yield better results in

terms of less area, better timing in hardware design. In our real design, we replace this multiplier with a shifter. This change will save hardware and also reduce delays by a few clock cycles. From computer simulation, we found that this modification does not affect the performance of the equalizer. Note that sign or sign-sign algorithms use only the sign bit of input signal or error signal in the tap update block, and thus can be used to further reduce hardware complexity. However, neither algorithm is used in this study because of performance loss associated with them.

After adding the necessary control signals to this block, we have the detailed design in Figure 3.9.

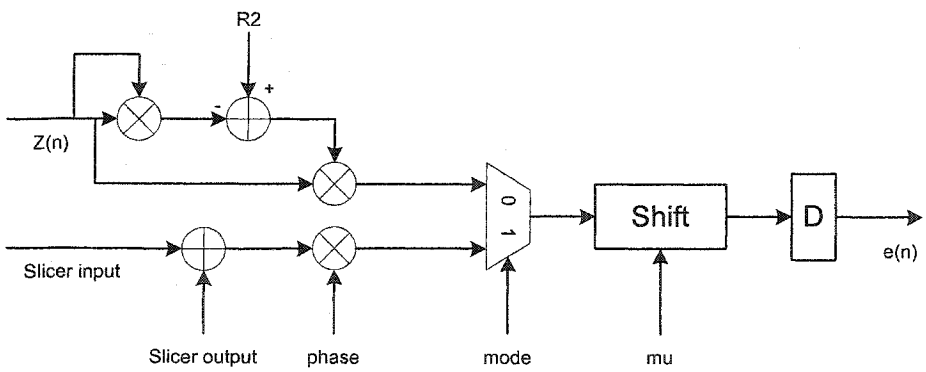

Figure 3.9 Diagram of Error block

We shall also indicate that except the 'mu' value is a real type and all other signals in these diagrams are complex signals.

3.3.3 Tap Update Block Design

The next block that will be considered is the tap update block. This block implements the function:

$$w(n+1) = w(n) + e(n) \cdot y^*(n) \tag{3.8}$$

The signal $w(n+1)$, $w(n)$ are coefficient vectors of next and current states. $e(n)$ is the error signal vector generated by the error block. $y(n)$ is the input signal vector.

The diagram of the tap update block is shown in Figure 3.10. The 'C1' is a control signal that controls the switch for hardware sharing. The 'C2' is another control signal that controls the coefficient update operation. C2 can control the coefficients update normally, or it forces the coefficients to a fixed value. One of its functions is to initialize coefficients in CMA mode. The reader may recall from the previous chapter that all coefficients should be fixed at zero, except the central coefficient, which is set to a high value in CMA mode to guarantee the equalizer converges to its global minima. A very important design part in this block is the delay by m function shown in diagram as 'MD'. Due to the pipelined nature of the high-speed equalizer design, the error signal that is arrived at the tap update block is really the result of input signals and coefficients M clock cycles ago. Therefore, to update coefficients, the input signals have to be delayed by the same number of clock cycles, so that the equation 3.8 will still hold. The 'M' has to be accurate. Even one clock cycles difference will cause equalizer function failure. This delay function is implemented by a shift register array in our design. One other option is implemented this delay function through a RAM (Random Access Memory) to save area when 'M' value is too large.

Except for the control signals, all signals in this block are complex.

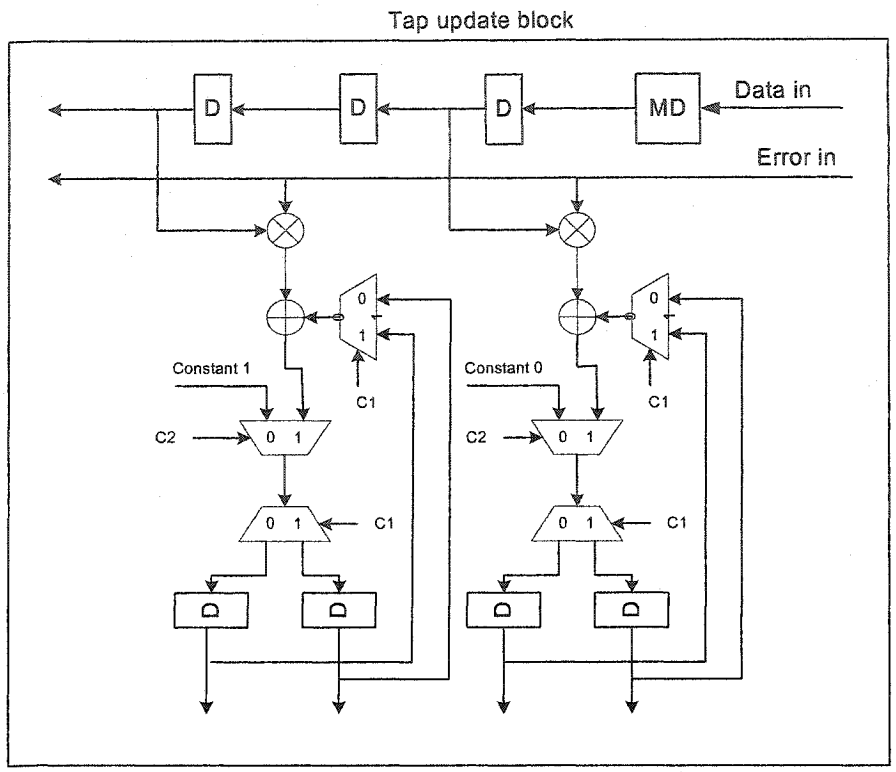

Figure 3.10 Tap update block diagram

3.3.4 Control Block Design

The control block consists of 3 sub blocks: lock detector, register file, and central control unit. Figure 3.11 shows the top level diagram of the control block.

The lock detector functions to detect lock and loss lock condition in both CMA and DD modes, so that the central control unit can control the equalizer action based on these lock or loss signals. This block is significant in the dual mode equalizer structure. It normally consists of a MSE calculator followed by a smoothing filter, a threshold comparator and a counter. A detailed diagram is not provided here because it is independent of the equalizer algorithm and can be built with different structure.

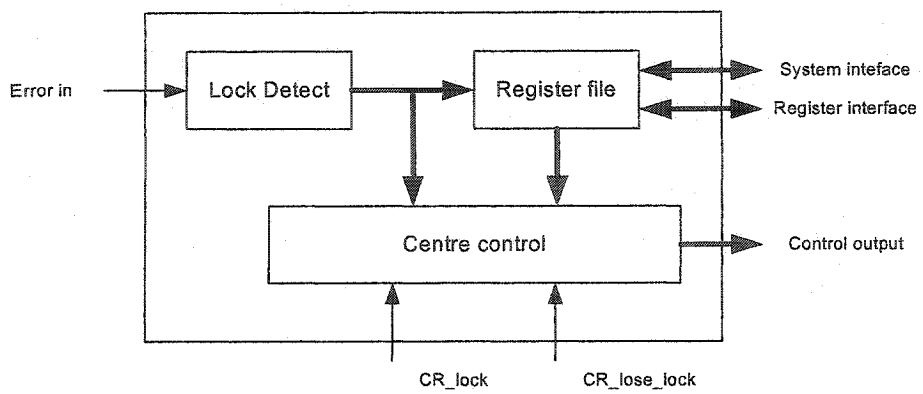

Figure 3.11 Top level diagram of control block

The register file is a place that holds the configurable parameters of the equalizer as updated step size, lock and loss lock thresholds, enabling signal and bypassing signals and some statistic parameters. This block has a system bus interface to communicate with a host processor, so that the host processor can configure its parameters, read its statistic parameters, complete control function on it. The bus interface is the same as other components in a modem design, and it is implementation dependent. Therefore, we will not discuss this block in detail. The central control unit mainly controls the mode switching between the CMA and the DD modes based on the information provided by the lock detector and by configuration register. This block also controls the switching function to implement the hardware sharing function.

The centre control unit consists of one FSM (Finite State Machine) and a few small logic control functions with counters. Figure 3.12 gives the state diagram of the main finite state machine. Its operation is self explained.

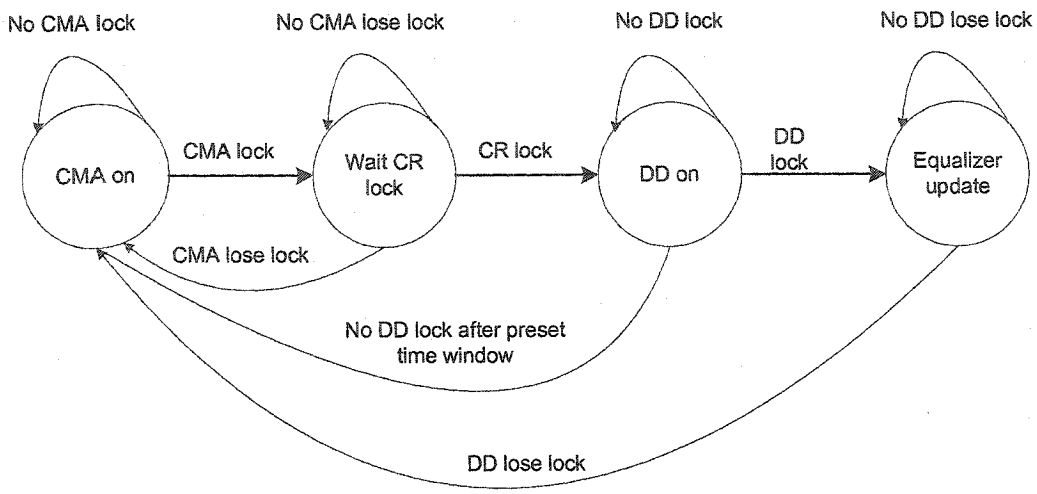

Figure 3.12 Main finite state machine state diagram in control block

3.4 Synthesizable VHDL Design

Before we discuss the VHDL design, we will present some design guidelines that will guide a VHDL design to be reusable, synthesizable, and technique independent.

When presenting these design guidelines, we assume the reader has a basic knowledge of VHDL. Otherwise, refer [59] to understand the basic of VHDL.

Design guideline 1- use following advanced VHDL features for reusable design:
Generics,
Generate statement,
Package of constant,
Record data type,
Component instantiation,
Attributes,
Configuration specifications,

Array aggregates,
Unconstrained array types,
Leaving unused output ports open.

Design guideline 2 – use only the synthesizable constructs of the VHDL. Many features in VHDL are used for simulation only; they should not be used for design. The following constructs should not be used for design:
TextIO package,
Time,
Real,
Multidimensional arrays,
Allies decorations,
Wait statement.

Design guideline 3 – use general statement for modeling functions and never instantiate technique dependent hardcore. If not avoided, use component instantiation.

Refer to [56, 58, 59, 60] for more details of the advanced VHDL features that can be applied for reusable, synthesizable, and technique independent RTL design.

Following are some examples of the blind equalizer design applying the design guidelines.

Figure 3.13 is a partial code of the blind equalizer configuration VHDL design. It demonstrates the use of package of constant and record data type. Our blind equalizer VHDL design is reusable by using constant to control all design parameters such as tap length, and component bit widths. All the constants are defined in a single place, which is to be found in the eq_config.vhd file. When designing this way, we will be

able to redesign the equalizer for different applications with different tap lengths, and different component parameters by modifying only the constants in the eq_config.vhd file. All main VHDL design codes will not be touched.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.all;


package eq_config is

constant IN_WIDTH: natural := 16;
constant OUT_WIDTH: natural := 16;
constant OUT_MSB: natural := 16;
constant OUT_LSB: natural := 1;
constant MAC_WIDTH: natural := 16;
constant MAC_MSB: natural := 16;
constant MAC_LSB: natural := 1;
constant PIPE_STAGE: natural := 6;
constant COEF_WIDTH: natural := 12;
constant COEF_MSB: natural := 12;
constant COEF_LSB: natural := 1;
constant STEP_SIZE: natural := 8;
constant TAP_LENGTH: natural := 10;
constant HALF_TAP_LENGTH: natural := TAP_LENGTH/2;

type in_type is record
     re_sig : std_logic_vector(IN_WIDTH-1 downto 0);
     im_sig : std_logic_vector(IN_WIDTH-1 downto 0);
end record;

type out_type is record
     re_sig : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
     im_sig : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
end record;

type mac_type is record
     re_sig : std_logic_vector(MAC_MSB-MAC_LSB downto 0);
     im_sig : std_logic_vector(MAC_MSB-MAC_LSB downto 0);
end record;

type coef_type is record
     re_sig : std_logic_vector(COEF_MSB-COEF_LSB downto 0);
     im_sig : std_logic_vector(COEF_MSB-COEF_LSB downto 0);
end record;

type coef_vector is array (natural range <>) of coef_type;

end package eq_config;
```

Figure 3.13 Use of constants and type

Figure 3.14 is the entity declaration part of equalizer FIR filter VHDL design. It defines the interface of the equalizer feed forward filter. Its entire interface signal is configurable by using generics and data types. All generics and data types used in here are defined in eq_config.vhd. In order to use the constant defined in eq_config, the design unit use clauses. "library work;" and "use work.eq_config.all" to make them visible in this design.

```
library IEEE;
library work;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;
use IEEE.std_logic_unsigned.all;
use work.eq_config.all;

entity eq_fff is
  generic(MUL_WIDTH   : integer;
          COEF_WIDTH : integer;
          ADD_WIDTH   : integer;
          OUT_WIDTH   : integer
          );
  port( clk : in std_logic;
        reset_n : in std_logic;
        ctrl1   : in std_logic;
        data_in : in in_type;
        coef_in : in coef_vector(0 to TAP_LENGTH -1);
        data_out : out out_type
        );
end eq_fff;
```

Figure 3.14 Equalizer FIR filter design entity

Figure 3.15 shows the Signal declaration in architecture. All signals are configurable by generics that are defined in eq_config.vhd. Another advanced VHDL feature used here is in the definition of 'out_add_type'. It is defined as a two-dimensional data array of unconstrained type. This data type is constrained by generic "HALF_TAP_LENGTH" when declared by a signal such as "out_add_re", "out_add_im".

```
architecture rtl of eq_fff is

   signal count : std_logic;
   type out_add_type is array (natural range <>)of
std_logic_vector(OUT_WIDTH-1 downto 0);
   signal out_add_re : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal out_add_im : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal reg_add_re : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal reg_add_im : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal coef_reg : coef_vector(0 to HALF_TAP_LENGTH-1);
   signal out_data_re : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
   signal out_data_im : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
   signal out_acc_re : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
   signal out_acc_im : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
```

Figure 3.15 Signal declaration in architecture

```
component eq_mac
  generic(MUL_WIDTH   : integer;
          COEF_WIDTH : integer;
          ADD_WIDTH   : integer;
          OUT_MSB     : integer;
          OUT_LSB     : integer;
          PIPE_STAGE : integer
          );
  port(clk : in std_logic;
    reset_n : in std_logic;
    x_re : in std_logic_vector(MUL_WIDTH-1 downto 0);
    x_im : in std_logic_vector(MUL_WIDTH-1 downto 0);
    y_re : in std_logic_vector(COEF_WIDTH-1 downto 0);
    y_im : in std_logic_vector(COEF_WIDTH-1 downto 0);
    in_add_re: in std_logic_vector(ADD_WIDTH-1 downto 0);
    in_add_im: in std_logic_vector(ADD_WIDTH-1 downto 0);
    add_re : out std_logic_vector(OUT_MSB-OUT_LSB downto 0);
    add_im : out std_logic_vector(OUT_MSB-OUT_LSB downto 0)
    );
end component;
```

Figure 3.16 Component declaration in architecture

Figure 3.16 is the component declaration in the architecture portion of the equalizer feed forward filter design. This figure combined with Figure 3.17 demonstrates the use of component instantiation and generate statement. Using the component instantiation makes the code reusable and technically independent, because the

detailed component is not in the main design and can be any detailed design or technology dependent design ware such as the Synopsis DesignWare component.

The generate statement used in Figure 3.17 shows the great flexibility of hardware generation controlled by generic. In this design, the generate statement handles the generation of filter taps. The generic "HALF_TAP_LENGTH" controls the number of taps and is defined in "eq_config"

```
mac_array : for i in 0 to HALF_TAP_LENGTH-1 generate
eq_mac0: component  eq_mac
 generic map (MUL_WIDTH => MUL_WIDTH,
              COEF_WIDTH => COEF_WIDTH,
              ADD_WIDTH => ADD_WIDTH,
              OUT_MSB => OUT_MSB,
             OUT_LSB => OUT_LSB,
              PIPE_STAGE => PIPE_STAGE
              )
  port map (clk => clk,
            reset_n => reset_n,
            x_re => data_in.re_sig,
            x_im => data_in.im_sig,
            y_re => coef_reg(i).re_sig,
            y_im => coef_reg(i).im_sig,
            in_add_re => reg_add_re(i),
            in_add_im => reg_add_im(i),
            add_re => out_add_re(i),
            add_im => out_add_im(i)
            );
end generate mac_array;
```

Figure 3.17 Component instantiation with generate statement

Appendix B shows the sample equalizer feed forward filter VHDL design that contains the more advanced VHDL features used in this equalizer design and it helps to better understand the content of this section.

3.5 Finite Word Effect and Parameter Selection

We have built a reusable, technique independent and synthesizable VHDL high speed blind equalizer model with all parameters configurable. An ASIC design can only be completed when all design parameters are fixed. In this section, we will first give a parameter selection method to fix the equalizer parameters. Then we will run RTL simulation using the reusable VHDL model. RTL simulation performs 3 functions, one is to verify that the parameter selection method is correct, and the second is to study the finite word effect on our proposed pipelined TDF equalizer. The results from the study will also serve as a final design parameter selection to complete our high speed equalizer ASIC design.

3.5.1 Equalizer Length Selection

There is no comprehensive and trustworthy method to estimate the number of taps required. Traditionally, we estimate the length of taps by the following:

1. Find a multipath model (FIR model), and invert it. Then count how many taps the invert model requires and then truncate the model since the inverted model will have infinite taps.
2. Find the most significant taps. Count the number of taps.
3. The equalizer taps will be 2 times (for 16 QAM) to 5 times (for 128 QAM) the number from step 2.

This estimation is based on the zero forcing principle. With the zero forcing principle, the equalization is obtained by inverting the multipath channel so that the overall effective spectrum will be flat. Based on this principle, the T/2 spaced equalizer will need double the number of taps that T spaced equalizer needs. However, from the previous discussion, we can observe that Zero-forcing equalization will be achieved using the T/2 spaced equalizer under the conditions:

1. Subchannels have no common roots and
2. Equalizer taps have at least the same length as subchannels.

Based on this method, the length of the T/2 spaced equalizer needs only to double the number of the multipath channel.

However some tests in real environment do not agree with the above theory [54]. Longer equalizer is still needed for good equalization. The author indicates that the zero-forcing condition met the fractional spaced equalizer requirement under the condition of ignoring the channel noise. When the noise is present, this zero-forcing equalization enhances the channel noise so that the equalizer needs more taps to compensate the enhanced noise.

Ye Li [53] indicated that there exist an optimal number of taps for specific step size. For the equalizer length other than this optimal number, the MSE will increase and may even diverge. This result is contradictory to the traditional equalizer theory. By increasing the number of taps, the equalizer should perform better. However this analysis is only applied to the blind equalization, and the convergence range is wider when the step size becomes smaller.

In our application, we use the blind equalization only at the acquisition mode. The equalizer is switched to DD mode after the equalizer has converged. At this level of MSE, the values of all taps of blind equalizer needed for reliability have a wide range. Therefore, we can still apply the traditional method to estimate the number of taps.

Based on the above analysis, we should set the number of equalizer taps to be more than double the channel length. As will be discussed in Chapter 4, the worse case channel model that we used is channel model 2 that has 4 taps. We chose the equalizer length to be 10 so that it can achieve optimal performance without using too much hardware. To verify the correctness of this length selection, we plot a RTL

simulation results in Figure 3.18 with different equalizer lengths. It shows that 10 taps are sufficient.

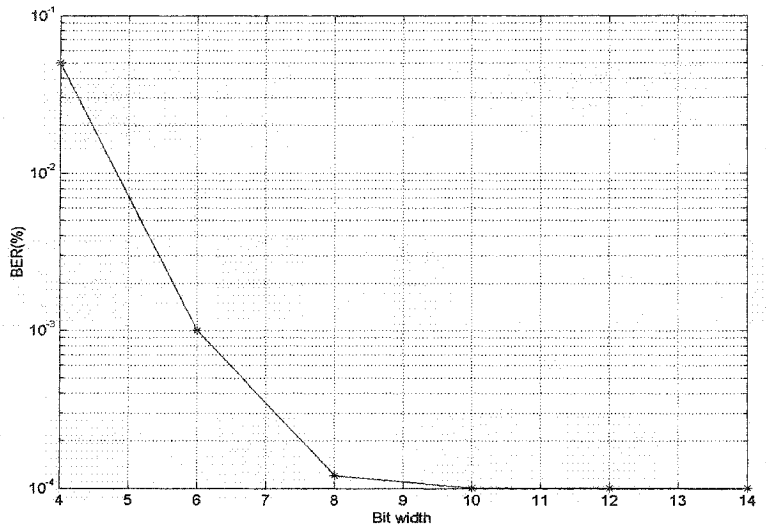

Figure 3.18 BER vs. number of taps

3.5.2 Component Word Length Selection

Finite word length is a very important issue in hardware design. If the word length requirement is not met, the equalizer performance will suffer by large implementation loss in terms of increased BER. On the other hand, if the word length of the equalizer is too large, the hardware complexity will increase and as a result the cost will increase. Another impact for the equalizer design having a large than necessary word length is the slowing down of speed. This is because the equalizer operation speed is dominated by the multiplier operation speed, which depends on word length. The wider the word length, the slower is the operation speed.

The word length requirement largely depends on the step size value. Therefore let us first select a proper step size before we estimate the word length requirement for the

equalizer. The step size is the only parameter that can be changed in the equalizer operation. The step size largely affects the equalizer performance.

The selection of step size depends largely on the equalizer structure. With a high-speed design objective in mind, the pipelined design will be the best choice. From the previous section we learn that the equalizer operation speed is limited by multiplication and addition. The speed of the multiplier is a function of the word length. A wider multiplier will need more pipelined stage to meet the speed requirements. As a result, the total loop delay will increase. Later we will show that the larger the pipelined delay, the smaller the step size should be. However, with small step size requirements, the word length requirements will increase. This looks like a positive feedback loop. To fix this problem, let us first assume that we will use 6 stages of pipelined multiplier and 1 pipelined delay for the adder. These numbers come from our rough analysis based on our speed requirement and our preliminary synthesis result. Based on this assumption, the total loop delay is approx 20. Based on the equation from [52], the step size should be:

$$0 < \mu < \frac{0.184}{\sigma_x^2(D+1)} = \frac{0.184}{10(20+1)} = 0.000915 \tag{3.9}$$

After selecting the step size, we tested the equalizer performance in terms of the BER with different step sizes based on the RTL model. Figure 3.19 plots the simulation results with different step sizes. It shows that the step size should be in the range of 0.00005 to 0.0001 to achieve the best performance and fast convergence speed.

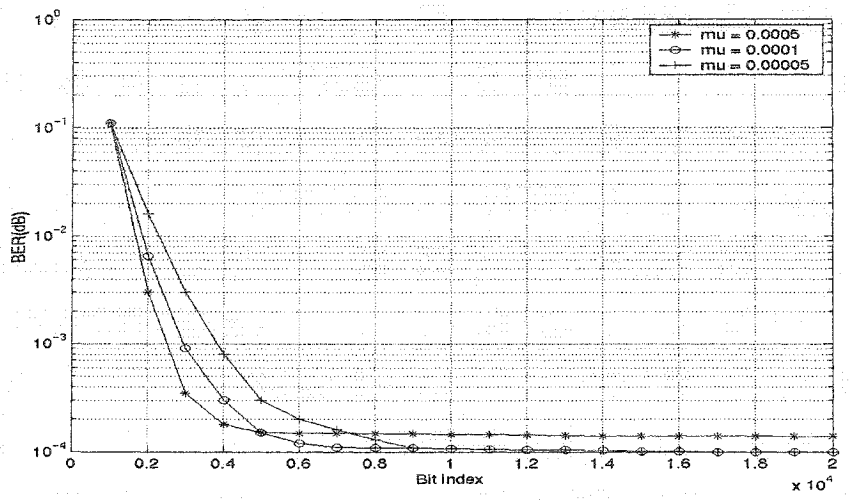

Figure 3.19 BER vs. step size

Now let us determine which parameter in the equalizer design is important and needs to be optimized. The equalizer essentially performs two operations. One is tap updating (equation 3.10) and the other is filtering (equation 3.11).

$$w^k(n+1) = w^k(n) + \mu \cdot e(n) \cdot y^*(n-k) \tag{3.10}$$

$$z(n) = \sum_{k=0}^{N-1} w^k(n) y(n-k) \tag{3.11}$$

From the above equation we found that there are four parameters that are important to the equalizer operation:

1. The size of the accumulator used for the tap coefficient updating. As the size of this accumulator decreases, the less the error information will be used to update its taps. This is because the product of error signal and step size will be of a very small value when the step size is small.
2. The size of the tap coefficient in the filter operation. This size should be smaller than the coefficient accumulator but should be big enough to keep its quantization noise lower than the channel noise. In the pipelined design, the

size of the tap coefficient will be kept large enough so that the update can operate when the error information, which is associated with the small step size, is very small.

3. The size of multiplier and accumulator used in the filtering operation to perform convolution. The selection of these sizes should be based on the consideration of the quantization noise observed at the equalizer output. The quantization noise accumulation effect due to the transpose form FIR filter structure should also be considered.
4. The bus width of the input signal. This size should also be optimized to limit quantization noise.

We use next procedure to calculate bus width:

1. Using our configurable VHDL equalizer design as device under test and run simulation with a selected channel model. Set all parameters as full 32 bits. Send a 30dB SNR QAM16 signal to the equalizer and check the output BER.
2. Reduce bus width of one parameter and keep other parameters at full precision. Run simulation with different bus width and get the output BER.
3. Repeat step 2 with other parameters until all 4 parameters are tested.

To select a suitable size for these mentioned parameters, the number of integer bits of all parameters should be calculated first.

To calculate the number of integer bit needed for the input signal, let us do some basic check. In our design, the equalizer is designed for 16 QAM applications. The maximum point at the constellation is (+/- 3, +/- 3). Considering the frequency offset, the signal will have the value $\sqrt{3^2+3^2} = \sqrt{18} = 4.5$. Adding multi-path signal to it, it will need 3 binary bits.

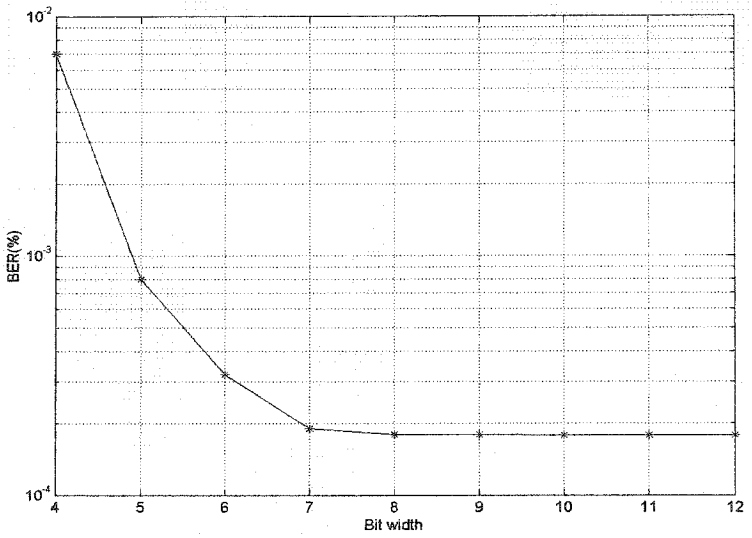

Figure 3.20 BER vs. bit width of input signal

Figure 3.20 shows the BER curve versus bit width of input signal. Its shows that a total 8 bits are sufficient for the input signal word length. We use the format <8,3,t> to represent the data type. It means that input signal needs total of 8 bits with 3 integer bits and the data type is two's complement format.

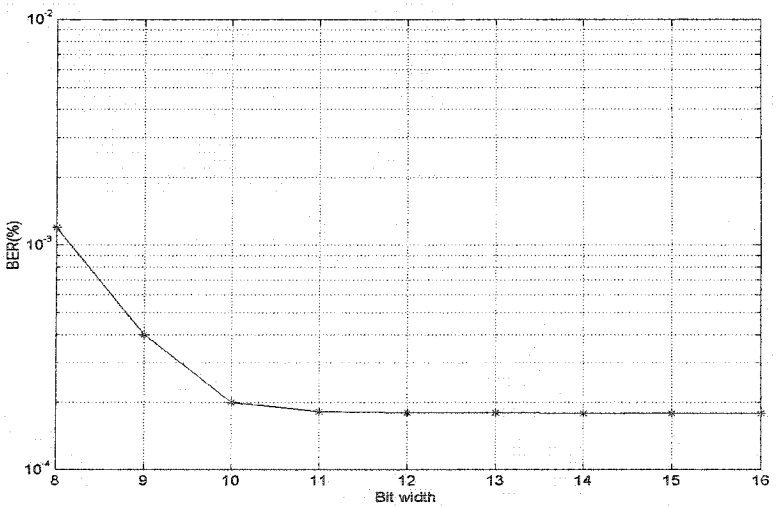

Figure 3.21 BER vs. bit width of tap update coefficient

Figure 3.21 plots the BER curve versus bit width of tap update coefficient. From the simulation result, we found that there was no overflow occurrence when the coefficient with 1 bit integer number. We can also see that at lower than 10 bits, the equalizer performance loses dramatically and above 12 bits bit width of coefficient, the equalizer performance has a negligible difference. Therefore the coefficient data format should be <12,1,t>. It simply means that the tap update coefficients need 12 bits total bit width and 1 integer bit.

Figure 3.22 plots the BER curve versus the bit width of the tap update coefficient. From the simulation result, we found that there was no overflow occurrence when the tap update accumulator has 1 bit integer number. We can see that at lower than 18 bits, the equalizer performance loses dramatically and this is because of the small step size we used. When the bit width is lower than 17 bits, the accumulator simply stops updating. When the bit width is greater then 21 bits, the equalizer performance has a negligible difference. Therefore, the coefficient data format should be <21,1,t>. It simply means that the tap update coefficients need 12 bits total bit width with 1 integer bit.

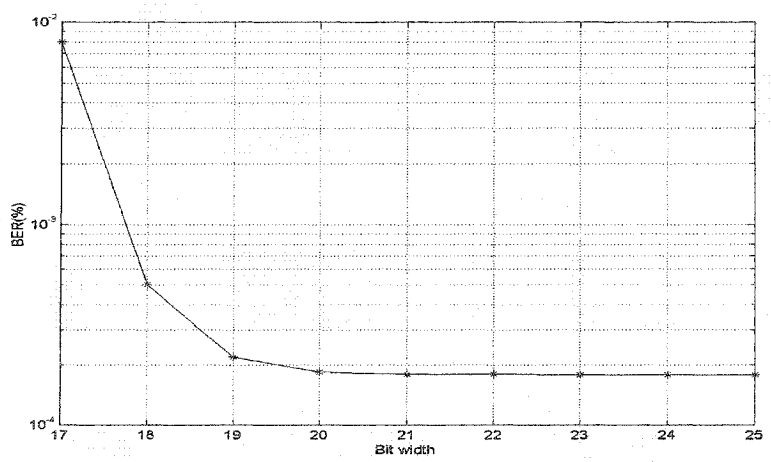

Figure 3.22 BER vs. bit width of accumulator in tap update block

Figure 3.23 plots the BER curve versus the bit width of filter MAC. From the simulation result, we found that there was no overflow occurrence when the filter MAC have 1 bit integer number. We can also see that when bit width of filter MAC is lower than 12 bits, the equalizer performance loses dramatically, and above 16 bits, the equalizer performance has negligible difference. Therefore the coefficient data format should be <16,1,t>. It simply means that 16 bits total bit width with 1 integer bit are needed for the filter MAC.

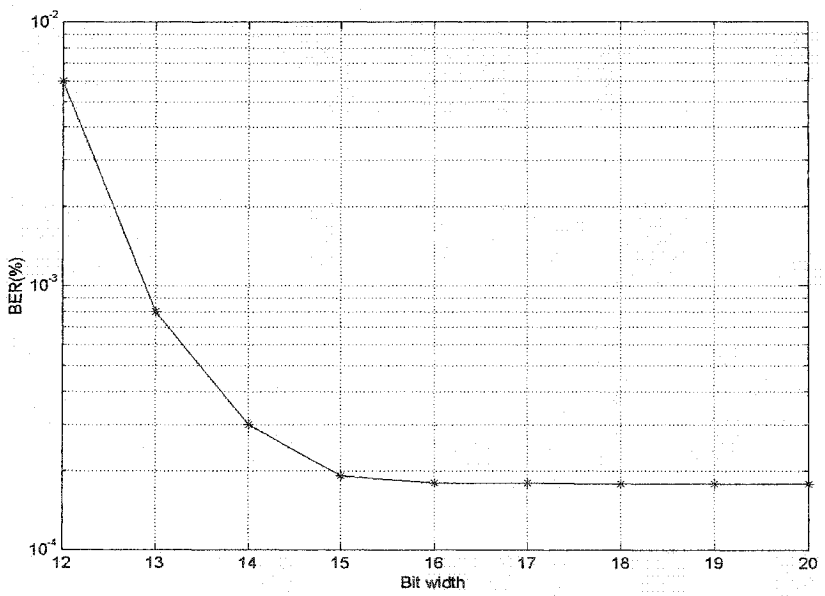

Figure 3.23 BER vs. bit width of filter MAC size

3.6 Summary

In this chapter, we discussed the detailed RTL implementation based on the high speed blind equalizer structure proposed in the last chapter. We also gave some useful design guidelines for how to design a reusable, synthesizable, and technically

independent RTL model using advanced VHDL features. These guidelines are further explained in detail when the design details are explained.

The selection of equalizer parameters is studied in depth. The criteria and methods of the parameter selection are provided with the proving of simulation results. These simulations serve as a study of finite word effect on the pipelined blind equalizer performance. It shows that with proper selection of design parameters, the pipelined TDF blind equalizer structure has a similar performance to a no pipelined behavior model.

4. Simulation

4.1 Test Bench Structure and Simulation Method

There are two test benches to be used in the test: a high level SPW test bench and a low level RTL test bench.

4.1.1 SPW Test Bench

The behavior model (floating point) simulations are performed using the SPW environment from Cadence. Figure 4.1 shows the system diagram of the test bench.

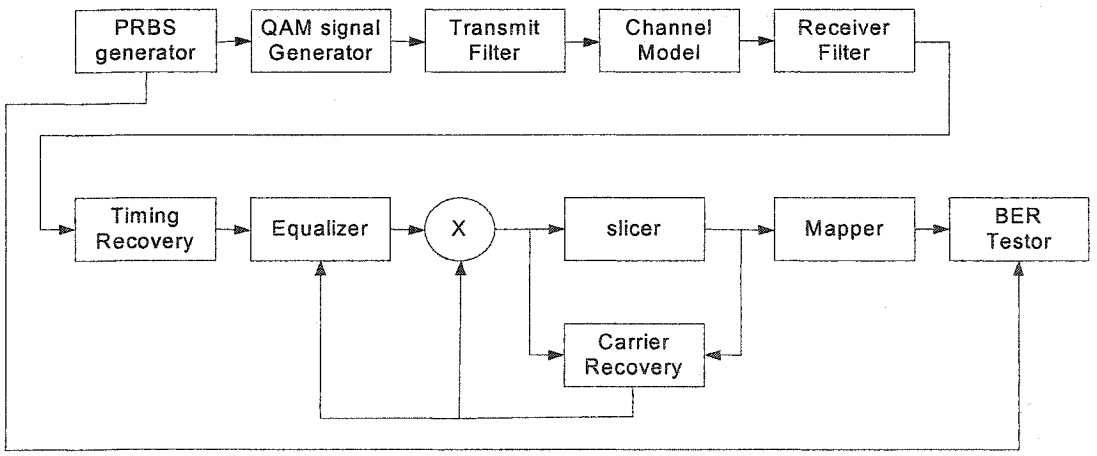

Figure 4.1 SPW simulation test bench

The PRBS generator is a programmable random signal generator based on the shift register structure. The QAM signal generator is simply a mapper that maps the incoming binary sequence to I (In phase) and Q (Quadrature) of a QAM signal. The

transmit filter is a SRRC (Squared Root Raised Cosine) FIR filter with a roll off factor of 0.25. This filter is for pulse shaping of transmission signal.

The channel model block generates clock jittering, carrier offset and add to the input signal. It also adds AWGN and multipath effect to the input signals.

Table 4.1 Channel model

Channel type	Tap number	Delay (ns)	Tap Amplitude	Phase (degree)
Model 1	1	0	0.995	0
	2	20	0.0995	-135
Model 2	1	0	0.286	-135
	2	20	0.953	0
	3	30	0.095	180
Model 3	1	0	0.804	-135
	2	3.6	0.581	180
	3	15.3	0.124	180

The multipath channel models listed in Table 4.1 represent different multipath conditions. Model 1 is a good channel with minor multipath effect. Model 2 is a bad channel having severe ISI. Model 3 represents the worst channel with a deep null in its frequency spectrum. Reference [57] will give more detail about these channel models.

The receiver filter is the same SRRC FIR filter as the one in the transmission side. Timing recovery is a timing control block that consists of a second order DPLL (Digital Phase Lock Loop) and an interpolator. A timing jitter of T/8 is set to verify the equalizer performance with no idea timing. Follow the equalizer, there is a complex multiplier, which is used for correcting the frequency offset controlled by

the carrier recovery circuit. The carrier recovery circuit is also a second order DPLL structure and its function is to find the frequency error. The slicer is a threshold signal detector and its function is to map the incoming signal to its nearest QAM constellation point. The mapper is used for mapping the sliced I and Q data to the QAM constellation. The BER tester has a comparator that compares the input data with the default PRBS sequence, has two counters that count the total processed data and the error data, and has a BER calculator to calculate the bit error rate based on the value of the two counters.

4.1.2 RTL Simulation Testbench

The purpose of this RTL simulation is to:

1. Verify the impact of the fixed point implementation on the performance of a blind equalizer.
2. Verify that the detailed hardware implementation matches with the floating point equalizer model in function and performance;
3. Provide information for the decision of design parameters. The information is the implementation loss associated with different design parameters. All design parameters are configurable and not fixed before this simulation stage.

The RTL simulation was performed using ModelSim simulation tools. The RTL test bench, as shown in Figure 4.2, is simpler than the high level test bench. The stimuli block simply read the data with added noise and other channel impacts from the SPW test bench, and fed it to RTL equalizer block. The complex multiplier, slicer and carrier recovery blocks are the similar function as the SPW test bench. The monitor block reads the output data from the slicer and then saves to a file. This file is then fed back to SPW to do the BER calculation.

It needs to be indicated that we only run RTL simulations without any timing check. No gate level simulation was conducted because we did not have any backend design information. However, it is enough to show that the hardware model functions correctly with the RTL level simulation result.

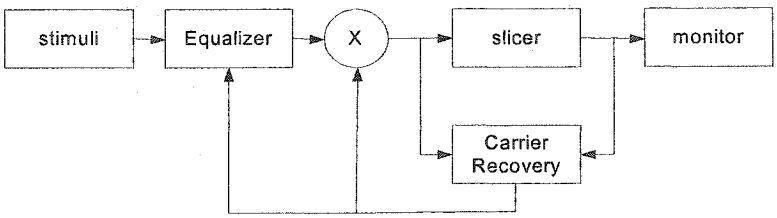

Figure 4.2 RTL simulation testbench

4.2 Test Result and Analysis

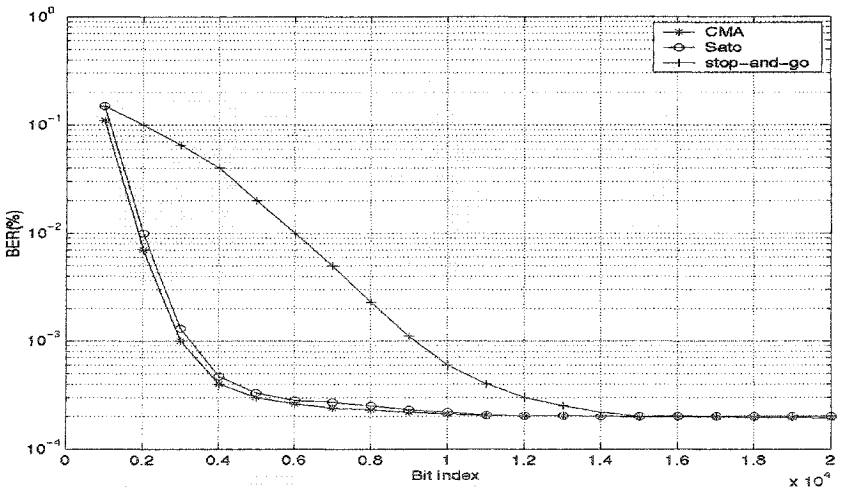

Figure 4.3 Learning curve of blind equalizers for channel model 2

Figure 4.3 plots the learning curves of the CMA, Sato, Stop-and-Go algorithms. All three algorithms have a tap length fixed at 10, and the applied channel model is channel 2. It is needs to be indicated that these curves are obtained by averaging BER of 1000 symbols and follow smoothing. The smoothing is necessary because the learning curve is still turbinated around the mean curve even after the average. The purpose of the learning curve plotting is the comparison of the converging speeds between the different algorithms.

From the learning curve we can see that the CMA and Sato algorithms have a similar converging speed, and the Stop-and-Go algorithm has the lowest converging speed.

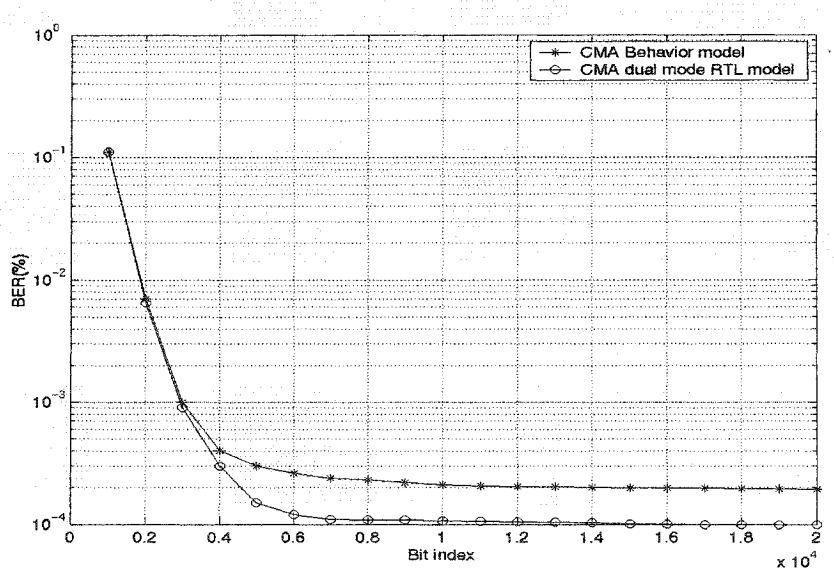

Figure 4.4 Learning curve of CMA equalizer for channel model 2

Figure 4.4 plots the learning curve of the CMA equalization for both the behavior and dual mode RTL models. It shows that the single mode CMA converges very fast at the beginning of process and the converging speed is very slow after about 3000 iterations. The remaining error is still significant because the constant modulus criterion. The dual mode CMA performance is much better than the single mode. It

uses the CMA mode first to quickly open the eye, and then switches to the DD mode to remove the remaining ISI quickly, and to do fine tuning.

Figure 4.5 and Figure 4.6 show the results of the BER test with the presence of different multipaths, a 5% carrier offset and noise. Symbol rate is 100Mbps. The clock jitter is T/8. The step size is 1.22e-4 for the CMA mode and 4.88e-4 for the DD mode. The equalizer has 10 taps for both the T/2 spaced fractional equalizer and the T spaced equalizer.

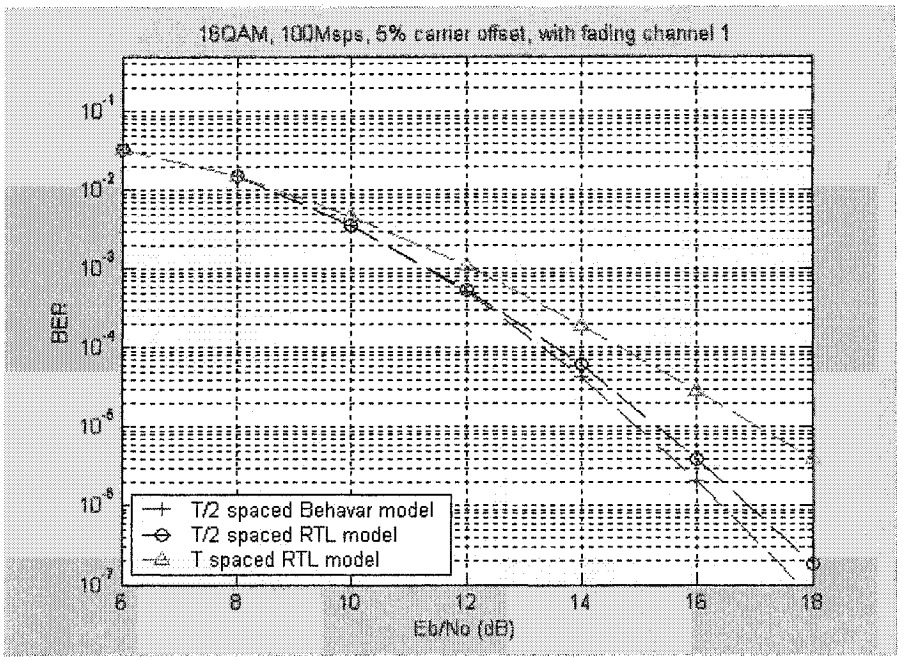

Figure 4.5 Simulation result with channel 1 with 5% carrier offset

From the BER simulation result we can see that the RTL model has ignitable loss compared to the behavior model(< 0.2dB).

We can see that the performance of using a T/2 spaced equalizer is at least 1dB better then the T spaced equalizer at a high SNR (>14dB) when the clock jitter is T/8.

We also need to indicate that the BER curve without the equalizer is the same as would be with a T spaced equalizer under a good channel. This result indicates that with good channel conditions and with a good timing recovery design, the use of the equalizer is not necessary.

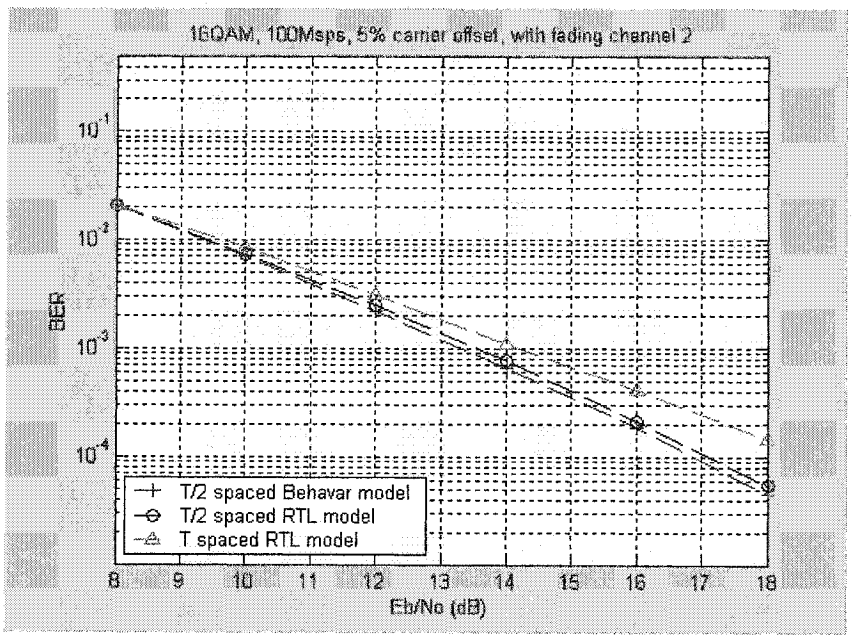

Figure 4.6 Simulation result with channel 2 with 5% carrier offset

In the channel 2 test, the signal cannot be recovered without using the equalizer. This result indicates that when the multipath effect dominates the channel conditions, the equalizer plays a key role in the operation.

Figure 4.7 and Figure 4.8 show the results of the BER test with 0% carrier offset. Other conditions remain the same as before.

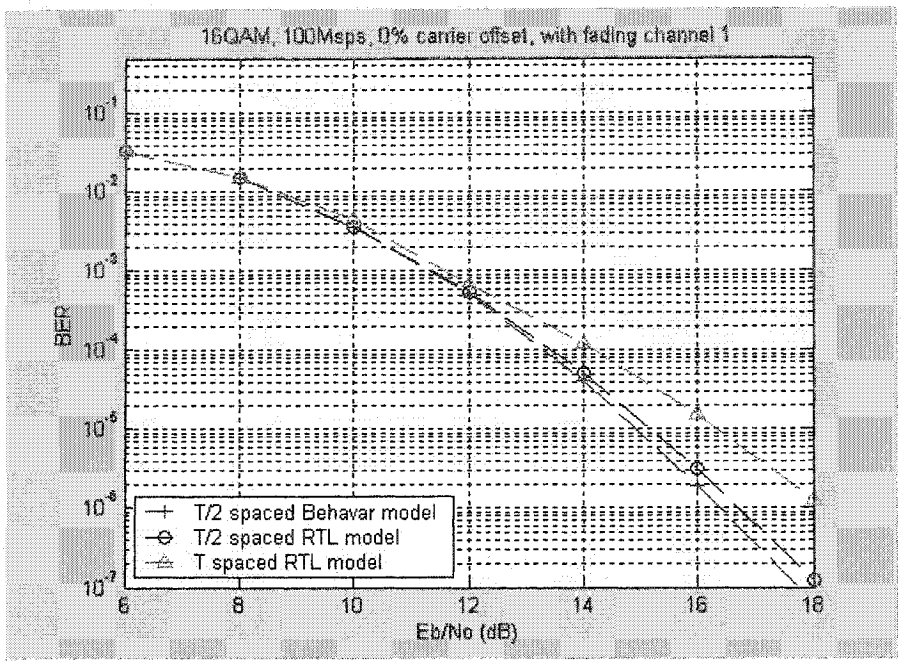

Figure 4.7 Simulation result with channel 1 with 0% carrier offset

Comparing Figure 4.7 with Figure 4.5, we found that the BER results show minor differences with the T/2 spaced equalizer. These 2 plots show results of the channel 1 simulation with and without frequency offset conditions. This indicates that with the help from the T/2 spaced equalizer, the performance of the carrier recovery circuit is almost independent of the carrier offset.

Comparing Figure 4.8 with Figure 4.6, we found that the BER results are almost the same as with the T/2 spaced equalizer. These 2 plots are results of the channel 2 simulation with and without frequency offset conditions. This indicates that under the severe multipath conditions, the ISI dominates the performance of the carrier recovery.

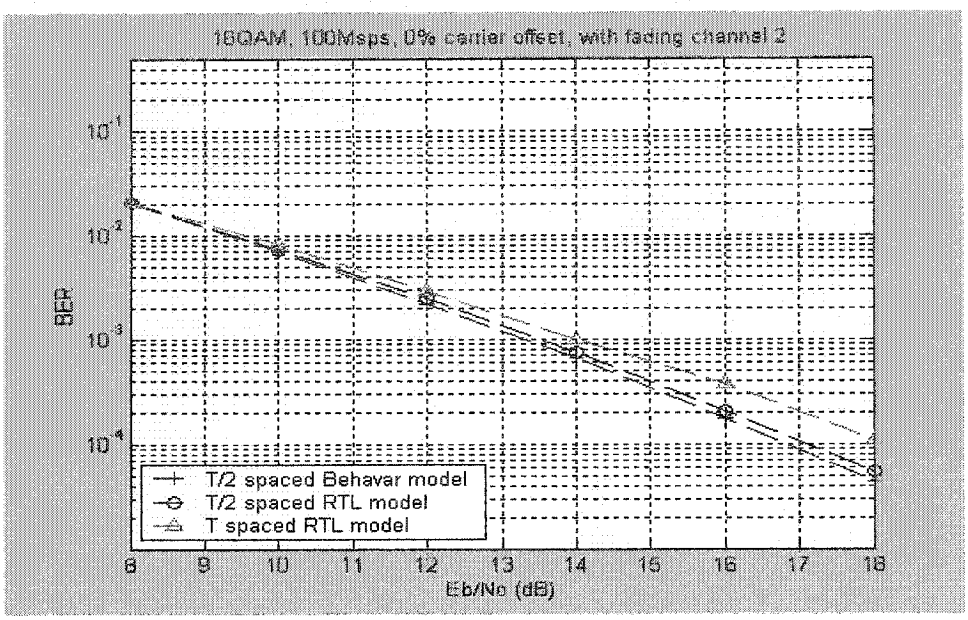

Figure 4.8 Simulation result with channel 2 with 0% carrier offset

It needs to be indicated that this equalizer does not work under channel model 3. This is because there is a deep null in the frequency spectrum of channel model 3. The linear equalizer cannot handle this severe channel condition.

Figure 4.9, Figure 4.10, Figure 4.11, and Figure 4.12 show the phasor diagrams for 16 QAM simulation results. The test is under the condition of no noise, 1% frequency offset and the multipath channel 2. These figures clearly show how the equalizer behaves. These figures correspond to the 4 phases in a procedure that were mentioned before. For convenience, we repeat the procedure here:

Phase 1. Initialization:

Set equalizer real center tap to 1 and other taps to 0. Reset Carrier recovery circuit. Set the equalizer to CMA mode without updating.

Phase 2. CMA mode:

Enable equalizer CMA mode to start updating. Carrier recovery circuit is still reset.

Phase 3. CR converging mode

When the equalizer CMA mode converges, turn on the carrier recovery circuit.

Phase 4. DD mode

When the carrier recovery circuit converges, switch the equalizer to DD mode.

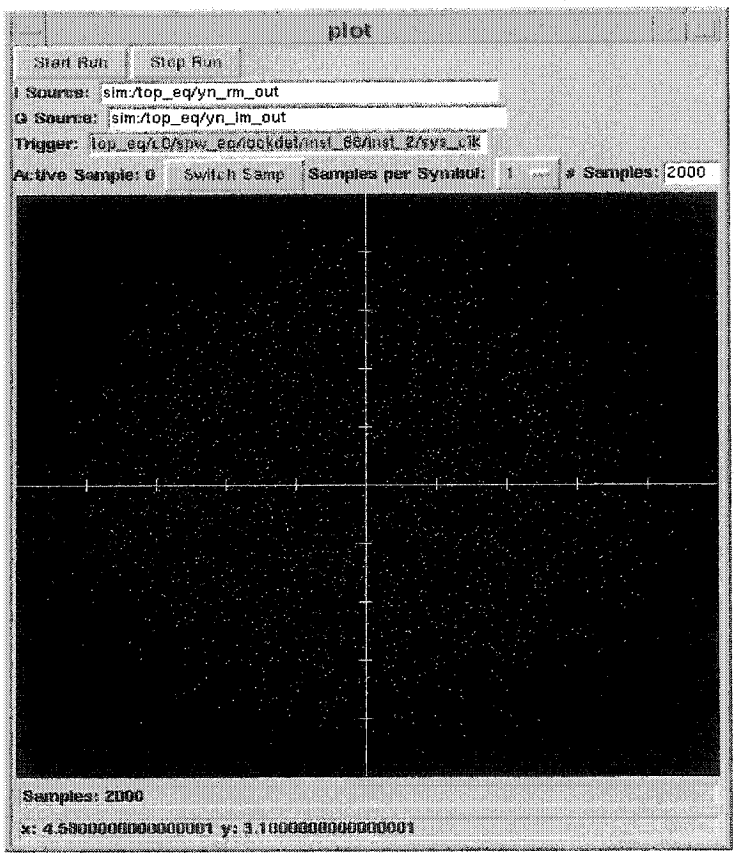

Figure 4.9 Signal constellation in equalizer initial phase

In Figure 4.9, the equalizer has not been turned on and the data constellation point can not be seen (the eye is completely closed).

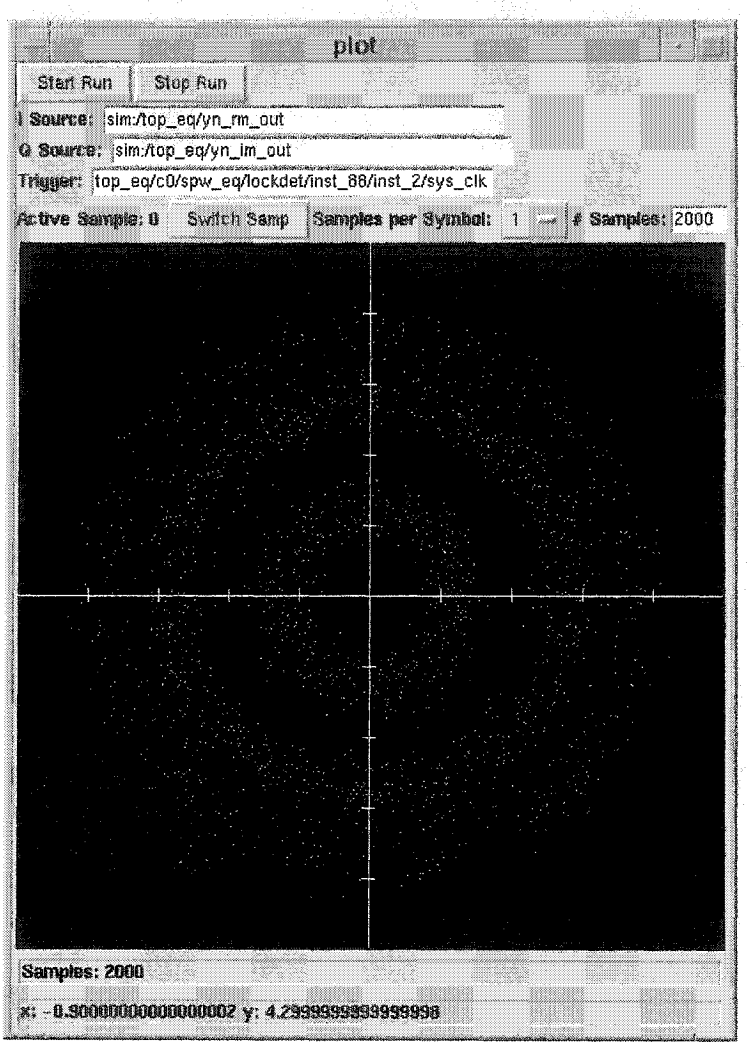

Figure 4.10 Signal constellation in equalizer CMA mode

In Figure 4.10, the equalizer CMA mode is converged but the carrier recovery circuit is not converged. We can see that 3 rings are in the constellation due to the frequency offset. (We will see a tilted 16-points constellation if only the phase offset is present and the frequency offset is not present).

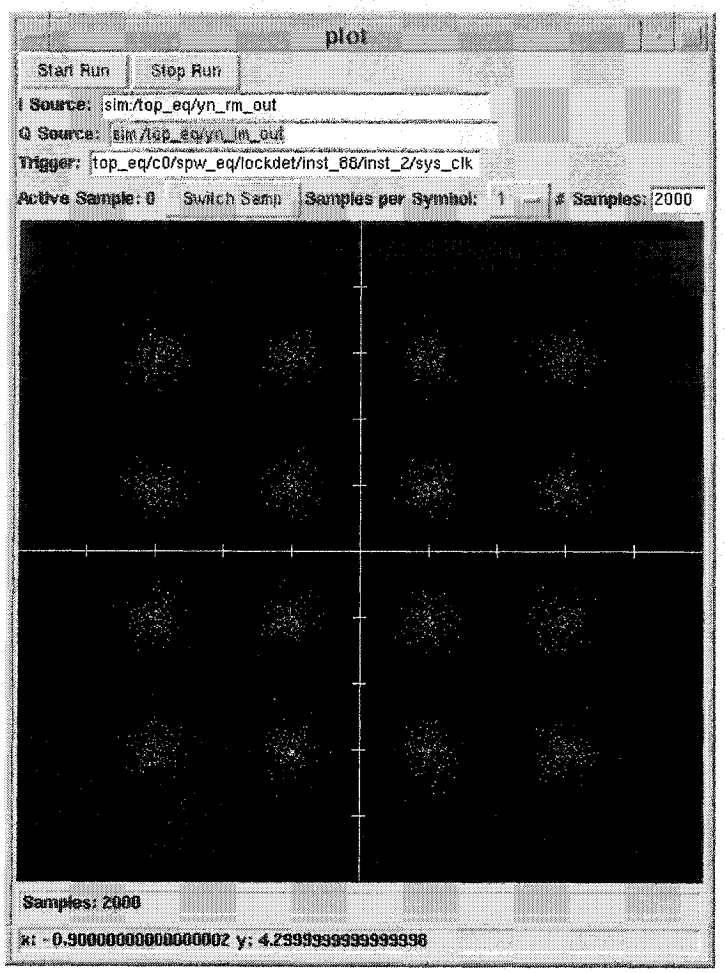

Figure 4.11 Signal constellation when CR convergence

In Figure 4.11, the Carrier recovery is converging so that we can see the 16-points constellation. However, the 16 points in the constellation are still large (the eye is open but not widely).

In Figure 4.12, the equalizer DD mode is converged and we can see 16 clean constellation points (the eye is completely open).

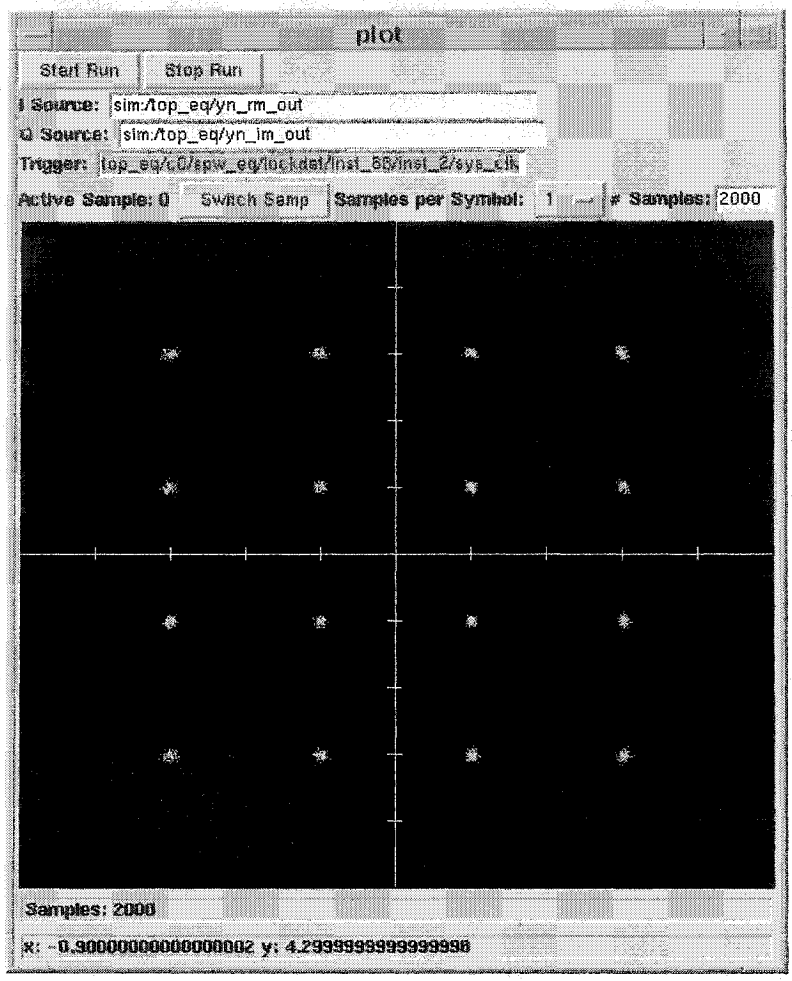

Figure 4.12 Signal constellation in equalizer DD mode

Simulation results show clearly that under severe channel conditions the carrier recovery circuit does not converge, even with a small frequency offset. After the equalizer removes most of the multi-path effects, the carrier recovery converges even with a large frequency offset (>5%)

Simulation results show that proper selection of the switching point from CMA to DD is very important. The selection of the switching point is decided by setting the threshold of the equalizer lock detector. The threshold is set at a larger value (for

example 0.1), so that the time to phase 3 is shortened (< 3000 iterations). The remaining MSE will still be large but shall be at the level that the eye is open enough to let the carrier recovery circuit converge. However, if the threshold is too large, the carrier recovery circuit will not converge, or even if the carrier recovery circuit converges the DD may not converge because the remaining MSE is too large. The step size for the DD mode shall be small enough so that, when switching from CMA to DD mode, the final remaining MSE is small.

Choosing the proper update step size for the CMA mode is even more important than the proper selection of the switching point. A large step size will lead to a fast converging time for CMA, but will increase the possibility of converging to a local minimum. Therefore we should keep the step size to a small value and should select a relatively large switch point threshold.

4.3 Summary

The behavior model test bench and the RTL model test bench were described. An SPW tool is used for building the behavior model test bench.

Simulation results of 3 different blind equalizer algorithms were presented and analyzed. These results show that all three algorithms have a similar performance but they have different converging speeds. The Stop-and-Go algorithm has the lowest converging speed.

Simulation results of a single mode and a dual mode CMA equalizer were presented and analyzed. The dual mode CMA performance is much better than the single mode, and it converges much faster.

BER test results for behavior and RTL models were presented with the presence of multipaths, carrier offsets, timing offsets, and AWGN. The simulation results from the behavior and the RTL modeling were compared and were found to be close to each other.

Simulation results also confirmed that a fractional equalizer performs much better than the T spaced equalizer.

The behavior of the proposed blind equalizer in different phases (CMA mode, CR convergence mode, DD mode) was illustrated by four snapshots from the RTL simulation results.

5. Logic Synthesis

In this chapter, we will discuss our proposed synthesis strategy and use the strategy to generate a gate netlist for our high speed blind equalizer ASIC. A static timing analysis will also be used to verify the speed of our equalizer design. Different components will be synthesized with different synthesis strategies. Synthesis results will be present and analyzed.

5.1 Proposed Bottom Up Synthesis Strategy

Our proposed synthesis strategy was built based on advanced Synopsis synthesis tool features, such as the data path synthesis, which use the Module Compilation within the Design Compiler option. These features enable the synthesis tool to optimally select the most suitable architecture based on the simple and high level component description. The advantage of using these features is that we need not design a low level component, and on the contrary, we need to design a high level description of the component and the synthesis tool will create a high speed and a low area design. This synthesis strategy can achieve optimal design results only by applying the reusable, synthesizable, and technically independent RTL design techniques on the RTL design entry.

This synthesis strategy consists of 2 steps. The first step is the synthesis of block level designs such as the FIR filter block or the tap update block. Then the top level (the entire equalizer) design will be synthesized with increased synthesis effort. This synthesis strategy is applied to our high speed blind equalizer design.

5.2 High Speed Equalizer Synthesis

Our synthesis tool is synopsis DC with MC (Module Compiler). The Synopsis design ware libraries are used for the basic components. The vender library selected is ST 0.25 μm HCMOS7_4. This design can easily be migrate to 0.18 μm or 0.13 μm techniques and other vender libraries.

The synthesis of some main components like multipliers and adders, were completed before the structural design due to the structural design needs rough estimation of the bus width, speed and pipelined stage number for those components. Parts of the synthesis results are shown in Table 5.1 and Table 5.2. The timing unit is ns for both tables.

The following is used to interpret the first column in Table 5.1:
"12 + 12 = 13" means that two 12-bit wide signals are added together to generate a 13-bits wide output signal.

Table 5.1 Timing for the adders

Bus width	Adder	Accumulator	Subtraction
12+12 = 13	2.25	2.36	2.33
16+16 = 17	2.56	2.51	2.47
24+24 = 25	2.60	2.58	2.53
32+32 = 33	2.83	2.76	2.80

The following is used to interpret the first column in Table 5.2:
"8 * 16 = 24" means that an 8-bit wide signal is multiplied by a 16-bit wide signal to generate a 24-bits wide output signal.

Table 5.2 Timing for multipliers

Pipeline / Bus width	2	3	4	5	6
8*12 = 12	4.87	3.53	2.90	3.29	2.34
12*12 = 12	4.90	4.09	3.25	3.04	3.08
8*8 = 16	4.86	4.26	3.35	3.00	2.98
8*16 = 16	5.54	4.48	3.46	3.39	3.34
16*16 = 16	5.72	4.81	3.84	3.50	3.42
8*16 = 24	5.42	4.62	3.94	3.84	3.01
16*16 = 24	6.55	4.75	3.82	3.64	3.46

It needs to be pointed out that these results are not optimized and are different from the final synthesis result. However, these results will help select components and step sizes during the structure design stage.

As indicated in these tables, the maximum speed the equalizer can achieve is less than 300M samples/s (timing > 3.4ns). Later we will show that with optimization, we can achieve a computation speed of 377M samples/s (timing < 2.7ns).

5.3 Synthesis results

The following are the synthesis results of the different components:

1. A complex tap, which consists of one complex multiplier and one complex adder pares with an 8-bit and a 16-bit input and a 16-bit output has:

   Total area: 287586 μm^2

   Timing: 2.60 ns

2. A complex multiplier with a 12-bit and a 16-bit input and a 16-bit output has:

   Total area: 342144 μm^2

   Timing: 2.57 ns

3. A complex multiplier with two 16-bit inputs and a 16-bit output has:

Total area: 97794 μm^2

Timing: 2.51 ns

4. An adder/subtract with two 16-bit inputs and 16-bit output has:

   Total area: 15012 μm^2

   Timing: 2.21 ns

5. A accumulator with 21-bit and a 21-bit input and a 21-bit output has:

   Total area: 33544 μm^2

   Timing: 2.46 ns

Block level synthesis results:

1. Error calculation block (in symbol clock domain):

   Tradition synthesis: Total area: 143523 μm^2, Timing: 4.54 ns

   Proposed synthesis: Total area: 121491 μm^2, Timing: 4.42 ns

2. Tap update block:

   Tradition synthesis: Total area: 1828760 μm^2, Timing: 2.71 ns

   Proposed synthesis: Total area: 1584370 μm^2, Timing: 2.57 ns

3. Filter block:

   Tradition synthesis: Total area: 1508470 μm^2, Timing: 2.81 ns

   Proposed synthesis: Total area: 1304832 μm^2, Timing: 2.65 ns

4. Control block:

   Tradition synthesis: Total area: 185466 μm^2, Timing: 2.73 ns

   Proposed synthesis: Total area: 167745 μm^2, Timing: 2.61 ns

Top level:

Tradition synthesis: Total area: 3920673 μm^2, Timing: 2.83 ns.

Proposed synthesis: Total area: 3219436 μm^2, Timing: 2.65 ns

The synthesis results show that the proposed synthesis strategy yields much better results compared to the traditional synthesis strategy. The final synthesis result shows that with the pipelined design, the critical path timing of the proposed equalizer structure is 2.65ns, which means the clock frequency can be running at 377MHz for the 0.25 μm IC technology.

5.4 Summary

A bottom up synthesis strategy was presented in this section. This synthesis strategy was built based on advanced Synopsis synthesis tool features such as the data path synthesis, which uses the Module Compilation within the Design Compiler option. This types of features enable the synthesis tool to optimally select the most suitable architecture based on simple and high level component descriptions.

The synthesis results of the proposed blind equalizer design were given. Using pipelined multipliers, the critical path timing of the proposed equalizer structure was 2.65ns. This timing information implies that the proposed blind equalizer can operate at a clock speed of 377 MHz for ST 0.25 μm IC technology.

6. Conclusions and Future work

6.1 Conclusions

Different blind equalizer algorithms were introduced and compared in terms of their speed, performance and robustness. The Constant Modulus algorithm was selected as the best algorithm candidate based on its performance and hardware complexity.

Different high speed hardware structures were introduced and compared based on their speed and hardware complexity. A pipelined transpose direct form structure was proposed as our high speed equalizer structure.

A detailed blind equalizer hardware structure based on the pipelined transposed direct form equalizer structure was implemented in VHDL. Some useful design guidelines for how to design a reusable, synthesizable, technically independent RTL model using advanced VHDL features were introduced. These guidelines were further explained in detail when the design details were explained.

The selection of equalizer parameters was studied in detail. The criteria and methods of the parameter selection were provided with the proven of simulation results. These simulations also served as a study of finite word effect on the performance of the pipelined blind equalizer. It shows that with the proper selection of design parameters, the pipelined TDF blind equalizer structure has a similar performance compare to the non-pipelined behavior model.

Behavior level simulation and RTL level simulations were conducted. Simulation results of 3 different blind equalizer algorithms were present and analyzed.

Simulation results of a single mode and a dual mode CMA equalizer were presented and analyzed. The dual mode CMA performance is much better than the single mode, and its convergence is much faster.

BER test results for the behavior and RTL models were presented with the presence of multipath, carrier offset, timing offset and AWGN. Simulation results from the behavior and RTL models were compared and were found to be close to each other.

Simulation results also confirmed that the fractional equalizer performs much better than the T spaced equalizer.

The behavior of the proposed blind equalizer in different phases (CMA mode, CR convergence mode, and DD mode) was illustrated with the use of four snapshots from the RTL simulation.

It has been shown, based on the synthesis results, that, with the proposed synthesis strategy, the blind equalizer implementation using pipelined transpose form structure can run at 377 MHz frequency. This speed means that the data throughput can reach up to 1.5 Gbps for the 0.25 μm IC technology.

6.2 Further work

As discussed in the thesis, the linear equalizer cannot handle severe channel conditions but the decision feedback equalizer can. Therefore, building a high speed blind decision feedback equalizer based on a pipelined transposed direct form structure can be considered.

More simulations with pipelined transposed direct form structure with different application and on different channel condition can be considered

Other technologies that could reduce gate count, area and power consumption of a blind equalizer can be considered.

Appendix A. Sample VHDL Code of Configuration

```
------------------------------------------------------
-- Author       : Zhonghe Song
-- Block name   : eq_config
-- Description  : This is the package that contain type
--              : definition and constants.
------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.all;


package eq_config is

constant IN_WIDTH: natural := 16;
constant OUT_WIDTH: natural := 16;
constant OUT_MSB: natural := 16;
constant OUT_LSB: natural := 1;
constant MAC_WIDTH: natural := 16;
constant MAC_MSB: natural := 16;
constant MAC_LSB: natural := 1;
constant PIPE_STAGE: natural := 6;
constant COEF_WIDTH: natural := 12;
constant COEF_MSB: natural := 12;
constant COEF_LSB: natural := 1;
constant STEP_SIZE: natural := 8;
constant TAP_LENGTH: natural := 10;
constant HALF_TAP_LENGTH: natural := TAP_LENGTH/2;

type in_type is record
     re_sig : std_logic_vector(IN_WIDTH-1 downto 0);
     im_sig : std_logic_vector(IN_WIDTH-1 downto 0);
end record;

type out_type is record
     re_sig : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
     im_sig : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
end record;

type mac_type is record
     re_sig : std_logic_vector(MAC_MSB-MAC_LSB downto 0);
     im_sig : std_logic_vector(MAC_MSB-MAC_LSB downto 0);
end record;

type coef_type is record
     re_sig : std_logic_vector(COEF_MSB-COEF_LSB downto 0);
     im_sig : std_logic_vector(COEF_MSB-COEF_LSB downto 0);
end record;

type coef_vector is array (natural range <>) of coef_type;

end package eq_config;
```

Appendix B. Sample VHDL Code of Feed forward Filter

```
-------------------------------------------------------
-- Author      : Zhonghe Song
-- Block name  : eq_fff
-- Description : This is the Feed Forward Filter block
-------------------------------------------------------
library IEEE;
library work;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;
use IEEE.std_logic_unsigned.all;
use work.eq_config.all;

entity eq_fff is
  generic(MUL_WIDTH   : integer;
          COEF_WIDTH  : integer;
          ADD_WIDTH   : integer;
          OUT_WIDTH   : integer
          );
  port( clk : in std_logic;
        reset_n : in std_logic;
        ctrl1   : in std_logic;
        data_in : in in_type;
        coef_in : in coef_vector(0 to TAP_LENGTH -1);
        data_out : out out_type
        );
end eq_fff;

architecture rtl of eq_fff is

   signal count : std_logic; type out_add_type is array (natural range <>)of
                  std_logic_vector(OUT_WIDTH-1 downto 0);
   signal out_add_re : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal out_add_im : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal reg_add_re : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal reg_add_im : out_add_type(0 to HALF_TAP_LENGTH-1);
   signal coef_reg : coef_vector(0 to HALF_TAP_LENGTH-1);
   signal out_data_re : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
   signal out_data_im : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
   signal out_acc_re : std_logic_vector(OUT_MSB-OUT_LSB downto 0);
   signal out_acc_im : std_logic_vector(OUT_MSB-OUT_LSB downto 0);

  component eq_mac
    generic(MUL_WIDTH   : integer;
            COEF_WIDTH  : integer;
            ADD_WIDTH   : integer;
            OUT_MSB     : integer;
             OUT_LSB     : integer;
            PIPE_STAGE : integer
            );
    port(clk : in std_logic;
      reset_n : in std_logic;
      x_re : in std_logic_vector(MUL_WIDTH-1 downto 0);
      x_im : in std_logic_vector(MUL_WIDTH-1 downto 0);
      y_re : in std_logic_vector(COEF_WIDTH-1 downto 0);
      y_im : in std_logic_vector(COEF_WIDTH-1 downto 0);
      in_add_re: in std_logic_vector(ADD_WIDTH-1 downto 0);
      in_add_im: in std_logic_vector(ADD_WIDTH-1 downto 0);
      add_re : out std_logic_vector(OUT_MSB-OUT_LSB downto 0);
      add_im : out std_logic_vector(OUT_MSB-OUT_LSB downto 0)
```

```
    );
  end component;

  component eq_add
    generic(ADD_WIDTH  : integer;
            OUT_MSB    : integer;
            OUT_LSB    : integer
            );
    port(clk : in std_logic;
      reset_n : in std_logic;
      in_re: in std_logic_vector(ADD_WIDTH-1 downto 0);
      in_im: in std_logic_vector(ADD_WIDTH-1 downto 0);
      add_re  in std_logic_vector(ADD_WIDTH-1 downto 0);
      add_im  in std_logic_vector(ADD_WIDTH-1 downto 0);
      out_re : out std_logic_vector(OUT_MSB-OUT_LSB downto 0);
      out_im : out std_logic_vector(OUT_MSB-OUT_LSB downto 0)
      );
  end component;
begin

mac_array : for i in 0 to HALF_TAP_LENGTH-1 generate
eq_mac0: component  eq_mac
 generic map (MUL_WIDTH => MUL_WIDTH,
              COEF_WIDTH => COEF_WIDTH,
              ADD_WIDTH => ADD_WIDTH,
              OUT_MSB => OUT_MSB,
             OUT_LSB => OUT_LSB,
             PIPE_STAGE => PIPE_STAGE
              )
  port map (clk => clk,
            reset_n => reset_n,
            x_re => data_in.re_sig,
            x_im => data_in.im_sig,
            y_re => coef_reg(i).re_sig,
            y_im => coef_reg(i).im_sig,
            in_add_re => reg_add_re(i),
            in_add_im => reg_add_im(i),
            add_re => out_add_re(i),
            add_im => out_add_im(i)
            );
end generate mac_array;

out_add_re(0) <= (others => '0');
out_add_im(0) <= (others => '0');

fff_out:  component eq_add
generic map(ADD_WIDTH => ADD_WIDTH,
            OUT_MSB => OUT_MSB,
           OUT_LSB => OUT_LSB
            )
port map(clk => clk,
      reset_n => reset_acc,
      in_re    => out_add_re(HALF_TAP_LENGTH-1),
      in_im    => out_add_im(HALF_TAP_LENGTH-1),
      add_re  => out_data_re,
      add_im  => out_data_im,
      out_re => out_acc_re,
      out_im => out_acc_im
      );

process (clk, reset_n)
  begin
      if reset_n ='0' then
        for i in 0 to HALF_TAP_LENGTH loop
          reg_add_re(i) <= (others => '0');
```

```
          reg_add_im(i) <= (others => '0');
        end loop;
      elsif rising_edge(clk) then
        for i in 1 to HALF_TAP_LENGTH loop
           reg_add_re(i) <= out_add_re(i);
           reg_add_im(i) <= out_add_im(i);
        end loop;
      end if;
end process;

process (clk, reset_n)
  begin
      if reset_n ='0' then
        for i in 0 to HALF_TAP_LENGTH-1 loop
          coef_reg(i).re_sig <= (others => '0');
          coef_reg(i).im_sig <= (others => '0');
        end loop;
      elsif rising_edge(clk) then
        for i in 0 to HALF_TAP_LENGTH-1 loop
           if count = '1' then
              coef_reg(i).re_sig <= coef_in(HALF_TAP_LENGTH + i).re_sig;
              coef_reg(i).im_sig <= coef_in(HALF_TAP_LENGTH + i).im_sig;
           else
              coef_reg(i).re_sig <= coef_in(i).re_sig;
              coef_reg(i).im_sig <= coef_in(i).im_sig;
           end if;
        end loop;
      end if;
end process;

process (clk, reset_n)
  begin
      if reset_n ='0' then
        count <= '0';
        out_data_re <= (others => '0');
        out_data_im <= (others => '0');
      elsif rising_edge(clk) then
        if count = '1' then
             count <= '0';
          out_data_re <= out_acc_re;
          out_data_im <= out_acc_im;
         else
           count <= '1';
          out_data_re <= out_add_re;
          out_data_im <= out_add_im;
        end if;
      end if;
end process;

data_out.re_sig <= out_data_re when count = '1';
data_out.im_sig <= out_data_im when count = '1';

end rtl;
```

References

[1] S.U.H. Qureshi, "Adaptive Equalization," *IEEE Proceedings*. Vol.73, pp. 1349-1387, Sept 1985

[2] A. Benveniste, "Blind Equalizers," *IEEE Trans. Commun.*, vol. com-32, pp. 871-883, Aug 1984

[3] J-J. Werner, "Blind Equalization for Broadband Access," *IEEE Communications Magazine*. pp. 87-93, April 1999

[4] K. Azadet, "Low-Power Equalizer Architectures for High-Speed Modems," *IEEE Communications Magazine*. pp. 118-126, Oct 1998

[5] B.R.Petersen and D, D, Falconer, "Suppression of Adjacent-Channel, Cochannel, and Intersymbol Interference by Equalizers and Linear Combiners," *IEEE Trans. Commun.*, vol. 42, pp. 3109-3118, Dec 1994

[6] B.C.W.Lo and K. B. Letaief, "Adaptive Equalization and Interference Cancellation for Wireless Communication Systems," *IEEE Trans. Commun.*, vol. 47, pp. 538-545, Apr 1999

[7] H. Bolcskei, "Blind Channel Identification and Equalization in OFDM-Based Multiantenna Systems," *IEEE Trans. Signal Procesing*. Vol.50, pp. 96-108, Jan 2002

[8] John G. Proakis, "Digital Communications", Third edition. McGraw-Hill 1995

[9] N. R. Shanbhag, "pipelined Adaptive Digital Filters," Kluwer Academic Publishers 1994

[10] B. Farhang-Boroujeny, "Adaptive Filters Therory and Applications". John Wiley & Sons 1998.

[11] D.N. Godard, ``Self Recovering Equalization and Carrier Tracking in Two Dimensional Data Communications Systems", *IEEE Trans. Commun.*, vol. COM28, No. 11, pp. 1867-1875, Nov., 1980

[12] G. J. Foschini, ``Equalizing without altering or detecting data," *Bell Syst. Tech.* J., vol. 64, pp. 1885-1911, Oct. 1985

[13] J. R. Treichler, B.G. Agee, ``A new approach to multipath correction of constant modulus signals", *IEEE Trans. ASSP.*, vol. ASSP31, pp. 459- 472, Apr. 1983

[14] Y. Sato, ``A Method of Self Recovering Equalization for Multilevel Amplitude Modulation Systems", *IEEE Trans. Commun.*, vol. COM23, pp. 679-682, June 1975

[15] G. Picci, G. Prati, ``Blind equalization and carrier recovery using a stop-and-go decision directed algorithm," *IEEE Trans. Comm.*, vol. COM35, pp. 877-887, Sept. 1987

[16] M. Goursat, A. Benveniste, G. Ruget, ``Robust identification of non-minimum phase system: blind adjustment of a linear equalizer in data communication," *IEEE Trans. Automat. Contr.*, vol. AC25, pp. 385-399, Jun. 1980

[17] B. Porat, B. Friedlander, ``Blind equalization of digital communication channel using higher order statistics", *IEEE Trans. on Signal Proce.*, vol. 39, no. 2, pp. 522-526, Feb. 1991

[18] S. Haykin, Editor, Blind Deconvolution, Prentice Hall, 1994

[19] L. Tong, G. Xu, T. Kailath, ``Blind identification and equalization based on second order statistics: A time Domain approach", *IEEE Trans. Inform. Theory*, vol. 40, no. 2, pp. 340-349, Mar. 1994

[20] E. Moulines, P. Duhamel, J. Cardoso, S. Mayrargue, ``Subspace methods for blind identification of multichannel FIR filters", *IEEE Trans. on Signal Proc.*, vol43, no. 2, pp.516-525, Feb. 1995

[21] Y. Li, Z. Ding, ``Blind channel identification based on second order cyclostationary statistics", in *Proc. ICASSP Conf.*, 1993

[22] J. K. Tugnait, ``On blind identifiability of multipath channels using fractional sampling and second order cyclostationary statistics", *Proc. Global Telecom Conf.*, pp. 2001-2005, Dec. 1993

[23] D. Boss, K. D. Kammeyer, ``Blind identification of mixed phase FIR systems with application to mobile communications channels", *Proc. ICASSP97*, vol. 5, pp. 3598-3592, Apr. 1997

[24] J. R. Trechler, et al., ``Practical Blind Demodulators for High Order QAM Signals", *Proceedings of IEEE*, Vol. 86, No. 10, pp. 1907 26, Oct. 1998,

[25] J. J. Shynk, R.P. Gooch, G. Krishnamurthy, C. K. Chan, ``A Comparative Performance Study of Several Blind Equalization Algorithms", Center for Info. *Processing Research UCSB, Applied Signal Technol.*, Sept., 1991

[26] C. B. Papadias, D. T. M. Slock, ``On the Decision Directed Equalization of Constant Modulus Signals", *Asilomar Conf.*, 1994

[27] Z. Ding, "Characteristics of band-limited channels unidentifiable from second-order cyclostationary statistics," *IEEE Signal Processing Letters*, vol. 3, pp. 150-152, mMay, 1996.

[28] Z. Ding, "On convergence analysis of fractionally spaced adaptive blind equalizers," *IEEE Trans on Signal Proc*, vol.45, pp. 650-657, march 1997.

[29] L. Tong, G. Xu, and t. Kailath, "Blind identification and equalization based on second-order statistics: A time domain approach," *IEEE trans on Information Theory*, vol. 40, pp. 340-349, March 1994.

[30] L. Tong, G. Xu, and t. Kailath, "Blind identification and equalization based on second-order statistics: A frequency domain approach," *IEEE trans on Information Theory*, vol. 41, pp. 329-334, January 1995.

[31] J.L. Hennessy and D.A. Paterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann, 1990.

[32] V. Carl Hamacher, Z.G Varnesie and S.Zaky, "Computer Organization" McGraw Hill publishing, 3rd edition.

[33] Neil Weste and Kamran Eshraghian, "Principles of CMOS VLSI Design", Addison Wesley publishing, 1988.

[34] Mauro Olivieri, "Design of Synchronous and Asynchronous Variable Latency Pipelined Multiplier", *IEEE transactions on VLSI systems* pp. 365-375, April 2001.

[35] Khuram Muhammad, Robert Staszewski and Poras Balsara, "Speed, Power, Area and Latency Tradeoffs in Adaptive FIR Filtering for PRML Read Channels ", *IEEE transactions on VLSI systems* pp. 42-51, February 2001

[36] Khuram Muhammad, Robert Staszewski and Poras Balsara, A 550M Samples 8-tap FIR digital filter for magnetic recording read channels", *IEEE J. Solid State Circuits*, vol35, pp.1205-1210, Aug 2000.

[37] Kamran Azadet and Chris J. Nicole, "Low Power Equalizer Architectures for High Speed Modems". *IEEE Communications Magazine*, pp.118-126, October 1998.

[38] Huy T. Nguyen and Abhijit Chatterjee, "Number Splitting with Shift-0 and-Add Decomposition for power and hardware optimization in Linear DSP synthesis". *IEEE transactions on VLSI systems* pp. 419-424, August 2000

[39] Simon Haykin, "Adaptiv Filter Theory", Upper Saddle River, N.J.: Prentice Hall, 1996.

[41] GUOZHU LONG, FUYUN LING, and JOHN G.PROAKIS, "The LMS Algorithm with Delayed Coefficient Adaptation:. *IEEE transactions on ASSP*. pp. 1397, Vol 377. No 8 September 1989

[42] M.D.Meyer and D.P.Agrawal, "A High Sampling Rate Delayed LMS Filter Architecture". *IEEE transactions on Circuits and Systems – II: Analog and Digital Signal Processing*. pp. 727. Vol. 40. No 11. November 1993.

[43] Markus Rupp and Rudi Frenzel, " Analysis of LMS and NLMS Algorithms with Delayed Coefficient Update Under the Presence of Spherically Invariant Processes". pp. 668. *IEEE transactions on signal processing*. Vol. 42.No 3. March 1994

[44] Rainer D. Poltmann, " Conversion of the Delayed LMS Algorithm into the LMS Algorithm". *IEEE signal Processing Letters*. pp. 223. Vol. 2. No. 12. December 1995.

[45] Ungerboeck, G. "Fractional Tap-Spacing Equalizer and Consequences for Clock Recovery in Data Modems," *IEEE Trans,Commun.*, vol. COM-24, pp. 856-864, August 1976

[46] Qureshi, S. U. H. and Forney, G. D., "Performance and Propertied of a T/2 Equalizer," *Natl. Telecomm. Conf. Record*. pp. 11.1.1-11.1.14, Los Angeles, Calif., December 1977.

[47] Gitlin, R. D. and Weinstein, S. B. "Fractionally-Spaced Equalization: An Improved Digital Transversal Equalizer," *bell Syst. Tech. J.*, vol. 60, pp. 275-296, February 1981

[48] Austin, M. E. "Decision-Feedback Equalization for Digital Communication Over Dispersive Channels," *MIT Lincoln Lab, Tech. Report*. No. 437 August1967

[49] Monsen, P. "Feedback Equalization for Fading Dispersive Channels," *IEEE Trans. Inform. Theory*, vol. IT-17, pp.56-64. January 1971.

[50] E. Moulines, P. Duhamel, J. –F. Cardoso, and Mayrargue,"Subspace method for the blind identification of multichannel FIR filters," *IEEE Trans on Signal processing*, vol. 43, pp. 516-525, Feburary 1995.

[51] Z.Ding, R. Kenedy, B. Anderson, and R. Johnson, "Ill-convergence of Godard blind equalizers in data communications systems," *IEEE Trans on Comm*, vol. COM-39, pp. 1313-1327, September 1991

[52] Douglas L. Jones " Learning characteristics of transpose-form LMS adaptive filters," *IEEE Trans on Circuit and systems – II Analog and digital signal processing*, vol. 39, No. 10, October 1992.

[53] Ye Li and K. J. Ray Liu, "Static and dynamic convergence behavior of adaptive blind equalizers," *IEEE Trans on signal processing*, vol. 44, No. 11, November 1996.

[54] J.R Treichler, I. Fijalkow, and C. R. Johnson, Jr.,"Fractionally spaced equalizers how long should they really be?" *IEEE Signal processing magazine.*

[55] Martin Vaupel, Heinrich Meyr, "High speed FIR filter Architectures with scalable sample rates,"

[56] Douglas J. Smith, "HDL chip design", Doone Publications 1996

[57] David Falconer, Wei Zhang, Nader Moayeri, "Specific Recommended Channel multipath Models for 802.16.1," IEEE 802.16.1pc-00/21 2000-04-24

[58] Ben Cohen, "VHDL coding styles and methodologies," Kluwer Academic publishers. 1999

[59] Peter J. Ashenden, "The designer's Guide to VHDL," Morgen Kanfmann publishers. 1996

[60] Synopsys, "Design compiler user guide," Synopsys. 2000.05

[61] C. Richard Johnson ..., "Blind equalization using the constant modulus criterion: a review," *IEEE proceedings*. Oct 1998.

[62] L. Hanzo, W. Webb, T. Keller, "Single – and multi – carrier quadrature amplitude modulation" John Wiley & Sons. 2000