Blind Adaptation of a Decision Feedback Equalizer for use in a 10Gbps Serial Link

By

Charles E. Berndt

B.Eng., Carleton University

A Thesis Submitted to the Faculty of Graduate Studies and Research in Partial

Fulfillment of the Requirements for the Degree of

Masters of Applied Science

Department of Electronics

Carleton University

Ottawa, Canada

January 2007

Your file Votre référence
ISBN: 978-0-494-26983-1
Our file Notre référence
ISBN: 978-0-494-26983-1

ABSTRACT

The Decision Feedback Equalizer (DFE) is commonly used in recovering data at the receive end of a multi-gigabit per second rate serial backplane channel. This form of equalization scheme typically relies on a training sequence. However this training sequence may not be shared by all implementations of the DFE, thus causing significant compatibility issues. A model of a DFE and adaptation algorithm was developed in MATLAB in combination with measurement data of a typical backplane channel. A method was developed to process the incoming signals from the channel involving a known adaptation algorithm in order to determine appropriate tap coefficients. These coefficients are shown to be effective for a data rate of 10 Gbps. The design presented would yield itself to standard-cell implementation except for the analog DFE structure and several stages of the up/down counters. Furthermore the design was evaluated using representative error sources and across several design parameters.

Acknowledgements

There are many people I would like to thank for their encouragement and support during the course of my degree. The first and foremost is my supervisor, Professor Tad Kwasniewski, for always guiding me with my best interests in mind and for creating an environment which is so ideal for learning. I would like to thank my research group, from which I have learned a great deal more than I had ever anticipated. To my friends in the department; John Danson, Harpreet Panesar, Vincent Karam, Victor Karam, Steve Penny, Travis Lovitt, Gord Allen, and Justin Abbott, who made the entire experience so much more enjoyable.

I would also like to thank my parents, who offered their endless love and support, which was always there to guide me. And to my sisters, who set the bar high, but were always there to give me encouragement along the way. Thank you.

Table of Contents

List of Tables

List of Abbreviations

CDR	Clock and Data Recovery
CML	Current-Mode Logic
DAC	Digital-to-Analog Converter
DFE	Decision Feedback Equalizer
DLL	Delay-locked loop
FEXT	Far-End Crosstalk
FFE	Feed-Forward Equalizer
FIR	Finite Impulse Response
FR4	Fire-Resistant 4-Layer Printed Circuit Board
IIR	Infinite Impulse Response
ISI	Inter-Symbol Interference
LMS	Least Mean Squared
LPF	Low-pass filter
NEXT	Near-End Crosstalk
NRZ	Non-Return to Zero
PAM	Pulse Amplitude Modulation
PCB	Printed Circuit Board
PLL	Phase-locked loop

List of Figures

CHAPTER 1 # *Introduction*

1.1 Introduction

The demand for ever greater high speed data communications has increased steadily for the past several years. This has led to an increase in data transmission rates within telecommunications equipment as well as consumer products. Therefore a great deal of effort has been focused on allowing greater and greater interconnect speeds within these types of equipment; leading to a variety of specialized techniques and circuit topologies.

Due to the complex timing issues and added manufacturing costs involved, simply increasing the number of parallel transmission paths has been ruled out as unfeasible. As a result, significant advances have been made in techniques which allow for higher speed data transmission over a serial backplane channel. These techniques are broadly categorized as channel equalization, since they have been created with the goal of preventing or reversing the negative effects caused by the limitations of the channel.

Modern high speed serial backplane channels include various circuit blocks whose exclusive function is to perform channel equalization. These circuit blocks can be implemented at either or both ends of the channel; at the transmitter as pre-emphasis, or at the

receiver as equalization. The pre-emphasis technique operates by modifying the transmitted bits in order to counteract the frequency response of the channel. The receive end techniques are more varied and in the case of *analog* and feed-forward equalization, operate very similar to the pre-emphasis technique by attempting to reverse the distortion caused by the channel. One very important technique is the decision feedback equalizer (DFE). This structure does not attempt to reverse the channel response, instead it focuses on removing the Inter-Symbol Interference (ISI) directly.

These various techniques, with the exception of the analog equalizer, are often implemented in the form of either a Finite or Infinite Impulse Response (FIR or IIR) filter. The various filter structures require their coefficients to be properly tuned in order to have a corrective effect on the data being transmitted. This tuning can be performed by calculations prior to implementation or while the filter is in operation. Although, selecting the coefficients before implementation may provide accurate results and reduce circuit complexity, it does not account for changing power and environmental conditions. Therefore an adaptive filter implementation is preferred. Unfortunately, an adaptive filter requires that the coefficient values must converge to the correct levels using an algorithm and a known sequence of data. This known sequence of data is transmitted across the backplane and the algorithm uses the incoming symbols to compare with the local copy. This comparison allows the algorithm to properly set the coefficients in order to minimize the amount of error at the output of the filter and is called filter training. Transmitting this known sequence and adapting the filter unfortunately wastes a significant amount of time which could be used for transmitting real data. Another major problem is that the training sequence used must be identical, and therefore creates significant compatibility issues.

1.2 Contributions

To date there have been many adaptive equalization schemes proposed in literature for use in a high-speed serial backplane channel. Many of these are CMOS implementations which typically employ a DFE structure capable of equalizing incoming signals. To do this effectively, the tap coefficients of these equalizers are usually trained at start-up using a specialized sequence of known data. The contribution of this work is a method for adaptivly tuning the coefficients of a 4-tap decision feedback equalizer which would be suitable for use at the end of a 34 inch serial backplane channel operating at 10 Gbps. It is the goal of this work to eliminate the training sequence and allow the adaptive DFE to converge its coefficients with the incoming data. Thus removing the dependence on proprietary or vendor specific technology. At the time of writing this document, there have been no detailed solutions reported.

The methodology used for this work was based entirely on simulations and experimentation. Since it would not be practical to develop a new algorithm for such a limited application using only mathematical methods, an alternative route was chosen. This route involved the use of the target application, in the form of a channel model, to revise an algorithm from previously developed work. The implementation of the proposed system is unnecessary as it would reside primarily in the digital domain which does not require silicon validation. Furthermore, the analog filter circuitry which would be required, has been quoted from literature. The stability of this solution has also been evaluated against the impact of possible mismatches typical of an analog DFE. While the methodology proposed in this work is based on a CMOS implementation, it is not necessarily limited to this technology.

1.3 Thesis organization

This document is organized into six separate sections. This first section provides a brief introduction to the topic and an outline of the contributions made to knowledge. The following section provides an in depth outline into the field of backplane signal integrity as well as a simplified overview of the various circuits used in adaptive equalization. Section three provides a full description of the research and accomplishments made in the method of blind adaptation of a DFE. The fourth section highlights basic details of how this method may be implemented into a real system and selected circuitry which would be required. This section also presents the power and area required in a similar implementation. Finally the conclusion will highlight the contributions made herein.

CHAPTER 2

Overview of Adaptive Equalization

2.1 Introduction

A significant amount of effort has gone into increasing the capacity of new serial backplane equipment through advances in the materials of construction and printed circuit board (PCB) design including techniques such as back drilling [1]. Unfortunately, the cost of replacing the existing equipment with these new types of backplanes is prohibitive. Therefore, there has been a great deal of focus on semiconductor based compensation circuitry which can allow for higher transmission speeds across both new and existing backplane channels. This chapter will discuss the channel, its signal integrity issues and the reasons why compensation circuitry is required. Next, a detailed description of the various circuits and techniques which are used to perform this compensation will be shown. This will include the topology, principles of operation, and implementation issues, followed by the requirements needed for them to operate effectively. Finally, there will be a brief overview of a state-of-the-art serial link topology including a brief discussion on how that topology effects the design of the adaptation engine. It is the goal of this chapter to give the reader an in-depth look into adaptive equalization and requirements that must be satisfied in a modern high-speed serial transmission environment.

2.2 The Serial Backplane Channel

The name serial backplane refers to a printed circuit board, on which many copper traces form independent serial channels between various daughter cards which are connected to a main board. These serial connections use gigahertz-speed data transmission in order to move large amounts of data from one card to another. Serial data transmission is used since the addition of many parallel transmission lines would significantly increase the cost of manufacturing the PCB as well as the chips and packages [3], [4]. It is for this reason, every effort should be made to increase the data transmission capacity of new and existing serial channels. Although a fiber-optic channel would be better suited for transmitting at such high data rates, the data must eventually pass through electrical equipment. Therefore, it is at the ends of these optical channels where serial backplane equipment is typically used in order to properly route information [2]. For the short length transmission between various parts of telecommunication equipment, or large scale computer equipment, the required overhead for optical transceivers would make such a system unfeasible.

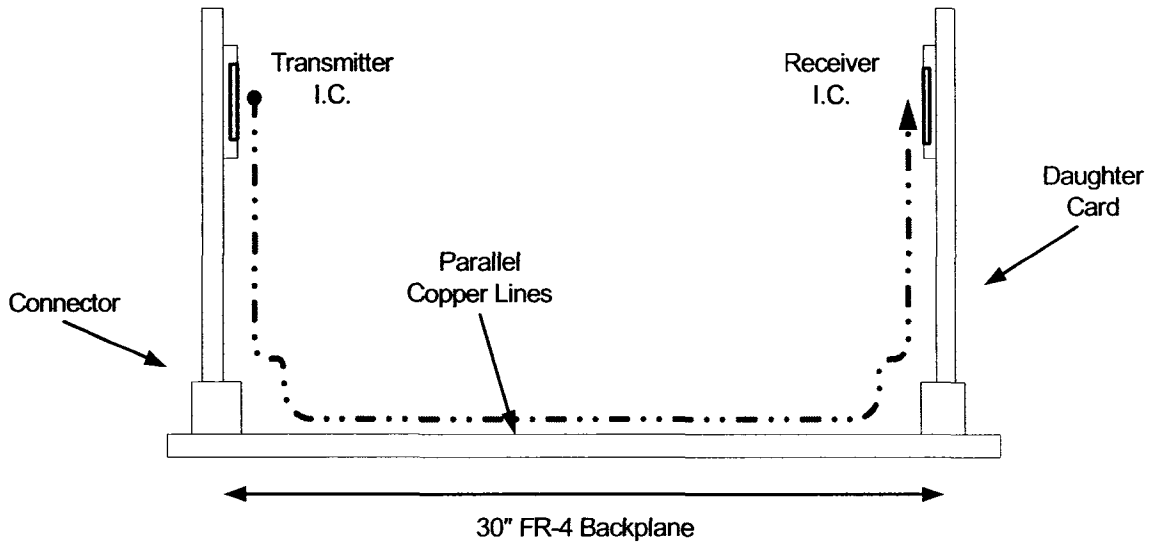

Figure 2.1 A Typical Backplane Structure.

Figure 2.1 above, illustrates a typical setup for a serial backplane channel with the signal path highlighted by the dotted line. The transmitter and receiver are on separate daughter cards connected through the copper traces on the surface of the backplane. Each daughter card may contain information processing hardware such as a switch, or other interface circuitry such as an optical transceiver.

2.2.1 Physical Description

The serial channel used for this work is a 30" TYCO Electronics, 10 Gigabit Attachment Unit Interface (XAUI) backplane with two daughter cards, each containing 2" of trace connected to the main board via two Z-Pack HM-Zd connectors. This standard evaluation channel is used to verify the inter-operability of equipment and circuits and is constructed using a standard type of fire-resistant 4 layer PCB (FR-4). This work utilizes electrical measurements of this channel, adopted from the IEEE standards committee, for use in testing serial transmission circuit designs [40], [42]. Unfortunately, the materials and connectors used in the backplane environment do not form a perfect, loss-less channel. These imperfections lead to significant problems when transmitting high-speed serial data [1]. In order to fully understand these issues, one must look at how the channel appears to the passing electrical signal, and how it is modelled. The following section examines these distortions, their source and further describes the electrical properties of the channel used in this research.

2.2.2 Electrical Performance

There are two major forms of distortion which are seen using a backplane channel, that is simple d.c. attenuation, and frequency dependant distortions. They are caused by the copper traces and energy storage within the connectors [1]. To properly understand

these frequency dependant distortions one must examine an typical model of the channel. Since the data signalling speed is in the gigabits per second range, the path along the back-plane, from the transmitter to the receiver, must be modeled as a distributed transmission line [4]. The exact point at which the channel can no longer be correctly modeled as a set of lumped passive elements and must be viewed as a distributed transmission line is beyond the scope of this document. However, a typical rule of thumb states that any transmission line resulting in a delay longer than 1/6 of the duration of the rising edge must be modeled using the distributed model [4]. Figure 2.2 below, shows an individual section of the distributed RLCG transmission line model. Each section contains 4 elements; the series self-inductance L, the series resistance R, the shunt conductance G and the shunt capacitance C. The series components are attributed to the copper trace along the length of the backplane, while the shunt elements represent effects from the dielectric material insulating the various layers of the PCB. Therefore the materials used in constructing the PCB represent a significant effect on the electrical performance of the channel.

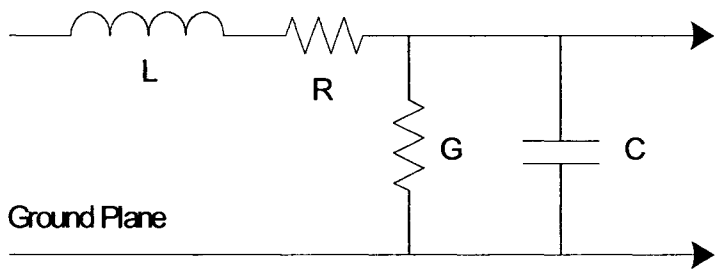

Figure 2.2 The L-R-G-C Transmission Line Model [4].

This model will also allow the designer to extract the propagation function (Eq. 2.1) of the transmission line as a function of the attenuation constant (α) and phase constant (β) [1],[4]. It can clearly be seen that these are both functions of the signaling frequency and the above model parameters. Further manipulation of this equation in order to solve

for either constant will yield two separate non-linear frequency dependant functions. Therefore, when a square-wave based data stream is sent across this transmission line, the various frequency components will see different attenuation and phase distortions [1].

$$\gamma = \alpha + j\beta = \sqrt{(R + j\omega L) \cdot (G + j\omega C)} \tag{2.1}$$

Further analysis of the model parameters within these functions will show that they themselves are also functions of frequency, due to the properties of the dielectric material within the PCB [4]. With these two attenuation constants the designer may compute the amplitude of a sinusoidal signal propagating down the transmission line using the following equation (2.2) [1]. It is clear from this equation that the various frequency components which make up the data signal would see different attenuation across the backplane channel.

$$A(x, t) = A_0 e^{-\alpha x} \cos(\omega t - \beta x) \tag{2.2}$$

To further understand these frequency dependant effects, one must examine the measured frequency response of the serial backplane channel. Figure 2.3 below, shows the frequency response from the serial backplane channel based on the measurements mentioned earlier [5]. From this graph, one can clearly see the low pass nature of the channel and the frequency dependant attenuation. This attenuation is especially severe beyond 1 GHz which unfortunately is where the desired baud frequency of the transmitted data will likely lie. Since the data being transmitted over the channel will be 2-Level Pulse Amplitude Modulation (PAM-2) or Non-Return to Zero (NRZ), the fundamental frequency of interest will be half of the data rate (assuming alternating ones and zeros). In this case, a

data rate of 5 Gbps has a fundamental frequency of 2.5 GHz which would see between 25 and 30 dB of loss.

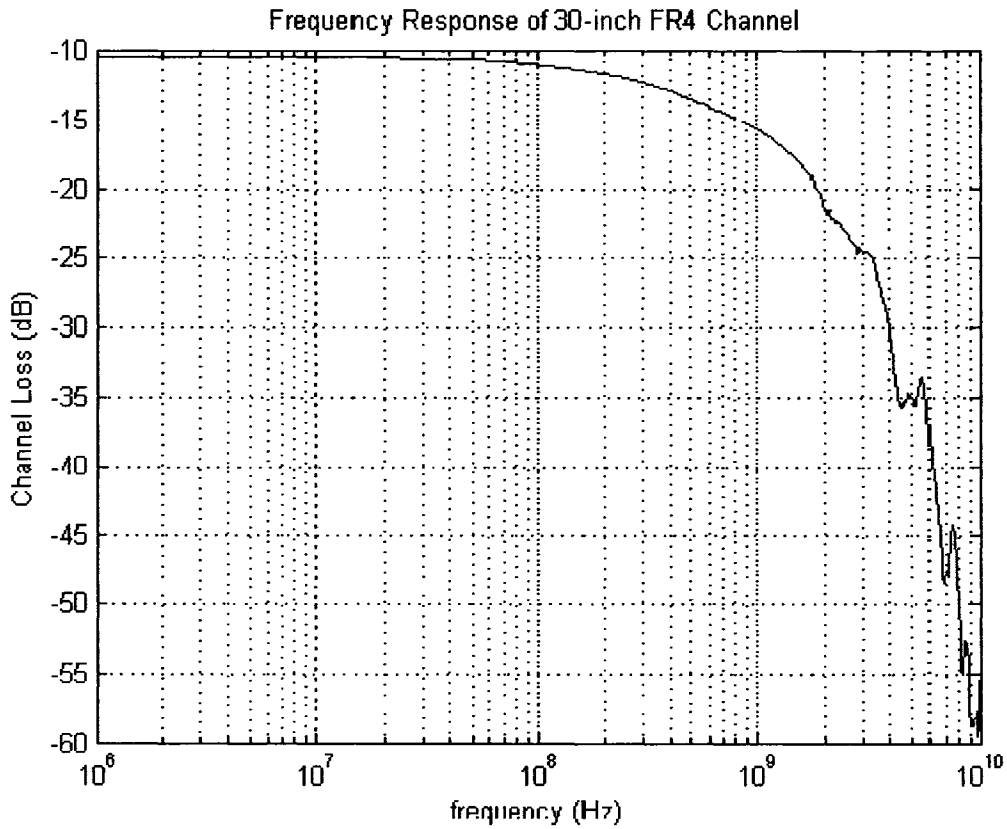

Figure 2.3 The Measured Frequency Response of the Serial Backplane [5]

2.3 Signal Integrity

Although the fundamental frequency will be half of the baud rate, the spectral components within the transmitted signal will vary according to the transmitted data pattern. As mentioned earlier, these various frequency components will see different attenuation across the channel. To better understand the effects of these various attenuations, one must look at the impulse response of the channel. Figure 2.4 below, shows the normalized

impulse response of the measured serial backplane channel used [5]. From this graph, the designer can extract several important properties of the channel.

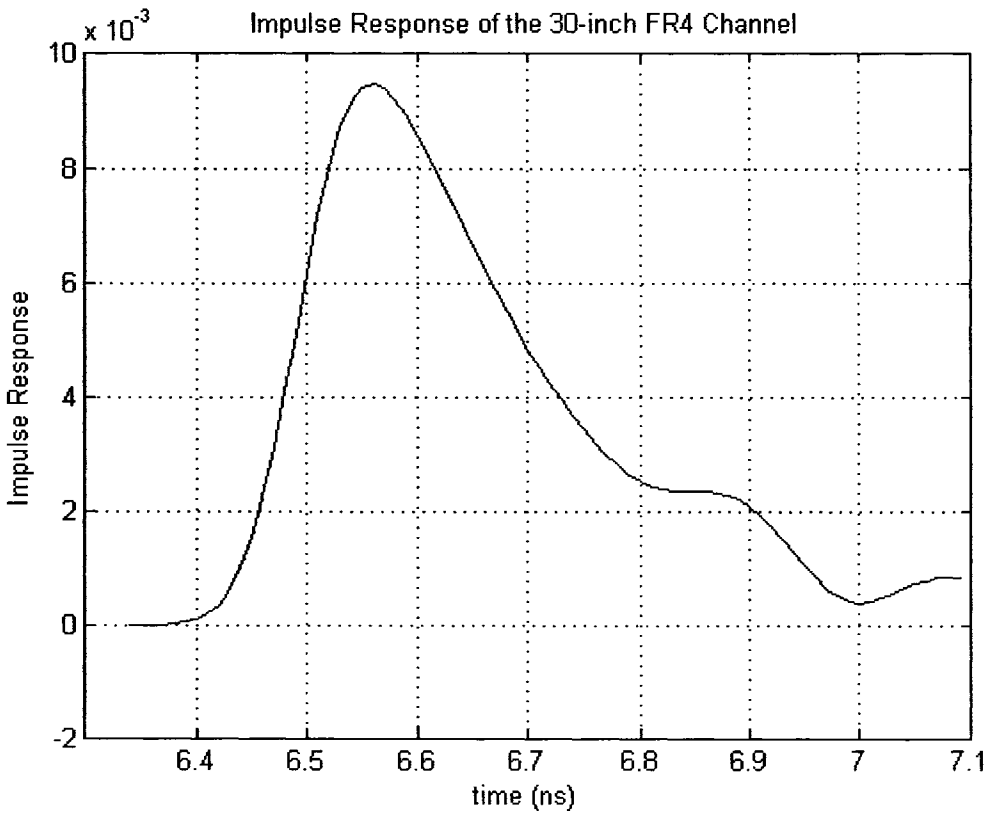

Figure 2.4 The Measured Impulse Response of the Serial Backplane [5].

One would first notice obvious information such as the channel delay, which gives the transit time required for any symbol to reach the receiver, and the overall attenuation of the channel. The next and more important information seen in the impulse response is the distortion in front and at the tail of the received impulse.

2.3.1 Inter-Symbol Interference

The distortion seen in front of the main peak (cursor) is called pre-cursor distortion and if severe enough, will interfere with the preceding data symbol. The long tail seen after the main peak is called post-cursor distortion. Because it lasts much longer and has a higher overall amplitude, the post-cursor distortion is considered the more severe of the two. However, as the data rate increases, the NRZ symbol period will shrink and the pre and post-cursor distortions will begin to overlap with the adjacent data symbols. This effect is called Inter-Symbol Interference or ISI. This interference from adjacent symbols becomes a very significant problem for which there must be compensation. Unfortunately, the low pass nature of the channel is not the only effect at work in distorting the transmitted data. Another major effect which can be seen in the impulse response of the channel, are reflections. In Figure 2.4 a reflection can clearly be seen by the characteristic amplitude increase at approximately 7.1 ns, this increase can interfere with data symbols which are several baud periods after the desired data symbol [6]. These reflections are caused by mismatches in the attenuation of the various components which make up the backplane channel, such as connectors, packages and bond wires [4]. These mismatches allow components to store energy temporarily, and release it to following data symbols, thus creating the undesired increase in amplitude [1].

It should be noted here that the impulse response and the frequency response do not show all of the sources of distortion seen across a serial backplane. These unseen sources include environmental effects, noise from external sources, voltage variations and most importantly, crosstalk.

2.3.2 Crosstalk

The serial backplane environment consists of many independent channels running in parallel over the entire length of the backplane. These independent channels can all be active simultaneously, furthermore these channels may even be directly adjacent to each other in order to reduce the manufacturing costs [1]. Therefore they would remain in close proximity through the chip package, along the daughter card, through the connectors and finally across the backplane. This results in relatively high coupling capacitance between the two channels. Figure 2.5 below shows the two forms of crosstalk which can drastically affect the signal integrity of the backplane channel.

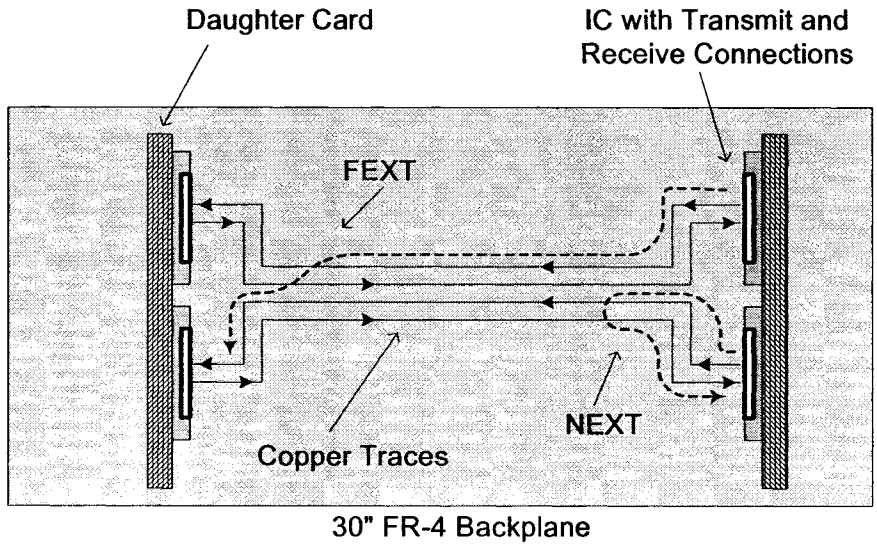

Figure 2.5 Overhead View of Backplane, Showing Crosstalk Paths.

The above figure shows an overhead view of a serial backplane channel. For simplicity, the diagram only shows two daughter cards and the copper traces linking them. In reality the channel consists of many more parts including packages, bond wires and card connectors, however the concept is unchanged by this simplification. The two types of crosstalk are distinguished by the relative positions of the offending transmitter and the

receiver. The first type, called Far-End Crosstalk (FEXT), occurs when a transmitted symbol couples onto a parallel path and is seen by a receiver at the same end as the desired receiver. FEXT is the weaker of the two types since the offending bit has been attenuated by the channel as well as the coupling process, therefore it appears as noise to the receiver [6], [3]. The second type of crosstalk is much more severe and is called Near-End Crosstalk (NEXT). This type of crosstalk occurs when symbols from an offending transmitter are coupled onto a channel whose receiver is in close proximity, possibly directly adjacent to the offender. Because of this proximity, the interfering symbol could have a drastically higher amplitude than the desired symbol [6]. Furthermore, this interfering symbol could be synchronized with the desired symbols which would completely destroy the desired information.

With the combined effects of the channel and ISI it is clear that in order to transmit data across the serial backplane channel at high speed, some form of compensation technique must be employed [7]. The remainder of this chapter will be dedicated to the various techniques and their implementation.

2.4 Channel Equalization Techniques

There are several techniques which can be used to compensate for the various distortions which are caused by the backplane channel. These techniques are all classified as channel equalization, since they attempt to compensate for the undesired channel characteristics. Initially used in data transmission over voice channels, the equalization techniques used in modern backplane and optical fiber channels have changed relatively little although the methods used in their implementation have changed drastically [6], [7], [8].

The ideal solution would involve a system which can completely compensate for all of the distortions due to the channel. This would involve a filter with a transfer function which is the inverse of the channel, however it would need to be infinitely long in the case of an FIR structure in order to remove all of the distortion [7].

It is for this reason that several different types of structures have been developed in order to combat the distortion, as opposed to inverting the channel. These structures or equalization methods fall into different categories depending on where they are located and how they operate. In terms of their location, equalizers are split into two groups. The first is called transmit pre-emphasis, which acts on the transmitted symbols before they enter the channel. The second group is called receive-end equalization and may contain several parts which process the received symbols after passing through the channel. Finally in the context of their operation, the equalizer structures are classified as either adaptive or fixed over time. These various types of equalization structures and their operation will be discussed in the following sections.

2.4.1 Transmitter Pre-Emphasis

As mentioned above, the strategy for a pre-emphasis circuit is to modify the outgoing data in order to correct for channel induced distortions. Unlike a forward error correcting technique, the pre-emphasis circuit does not add redundant information to the outgoing symbols for later recovery. Instead it adds redundant power to the higher frequency components in order to overcome the low pass nature of the channel [9]. The pre-emphasis technique also uses the symbols which have previously been transmitted in order to counteract their ISI contribution [7]. This is accomplished through the use of a linear transversal filter also known as a finite impulse response filter at the transmitter. The filter, as seen in figure 2.6, is basically a tapped delay-line which is connected in a

feed-forward configuration. For each baud period a new data symbol enters the filter and previous data passes through the next delay element. The total number of which varies according to the length of the filter.

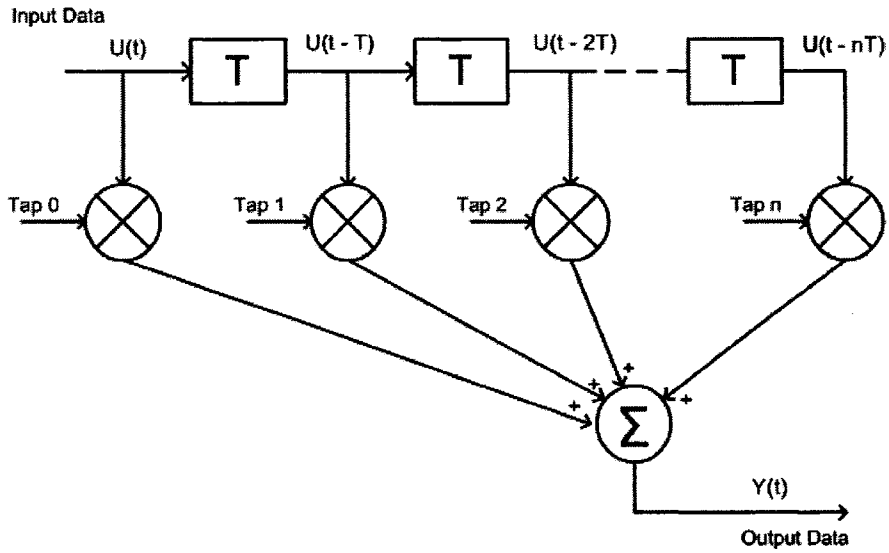

Figure 2.6 The Transmitter Pre-Emphasis Circuit

Each of these symbols is multiplied by an individual tap value according to the desired proportion typically determined by their individual ISI contribution [7]. These proportions are important to the effectiveness of the filter, and their values will be discussed in further detail later in this chapter. The overall output symbol will be based on the proportional sum of the current and past information as described in equation 2.3.

$$Y(t) = \sum_{n=0}^{N} Tap(n)U(t-nT) \tag{2.3}$$

Pre-emphasis circuits are a relatively popular method for combating ISI because of their ease of implementation [9], [10]. This technique sometimes falls under the name de-emphasis which simply indicates that the polarity of the coefficients are such that the rising edges of the data symbols may have a purposefully reduced initial amplitude. Unfortunately there are significant penalties with the use of the pre-emphasis technique.

The first is that it relies on added voltage swing in order to compensate for the high-frequency attenuation of the channel. However the output is still limited and therefore the effectiveness of the technique is also limited. Furthermore, the added high-frequency power serves to enhance both near and far-end crosstalk [9],[10]. Another limitation of the pre-emphasis technique is that there is no inherent feedback from the receiver in order to verify its effectiveness. Therefore, the tap coefficients are fixed and a great deal of effort must go into their selection during the design [10]. Alternatively some form of feedback path from the receiver can be created in order to control the coefficient values which would consume additional resources and time [5], [11].

2.4.2 Receiver Equalization

An alternative strategy for correcting the ISI is to recover the data on the receive-end of the serial backplane channel. Recovering the data in the receiver involves using a filter to remove the distortion added by the channel, then making a decision on the value of that filtered data. This filtering usually involves two parts, a linear feed-forward equalizer and a non-linear decision feedback equalizer (DFE) [12]. The linear forward equalizer is very similar to that found in the transmitter pre-emphasis technique, however it does not need to completely reverse the channel response. Rather, it is specifically targeted at the

pre-cursor ISI, which involves symbols in the future with respect to the current data symbol. Therefore it does not specifically enhance the high-frequency noise components of the incoming symbols, causing an increase in crosstalk like the pre-emphasis circuit. Any noise seen at the input is processed and added to any subsequent symbols [3], [7], [12]. This technique does however reduce the burden placed on any subsequent circuits such as the DFE.

The second circuit employed at the receiver is the decision feedback equalizer. This is most effective at removing post-cursor ISI since it utilizes previously detected symbols to remove their contribution to the interference [7], [12]. This is accomplished using a structure similar to that of a infinite impulse response (IIR) filter, with a built in decision device. It is because of this decision that the DFE is called a non-linear transversal filter.

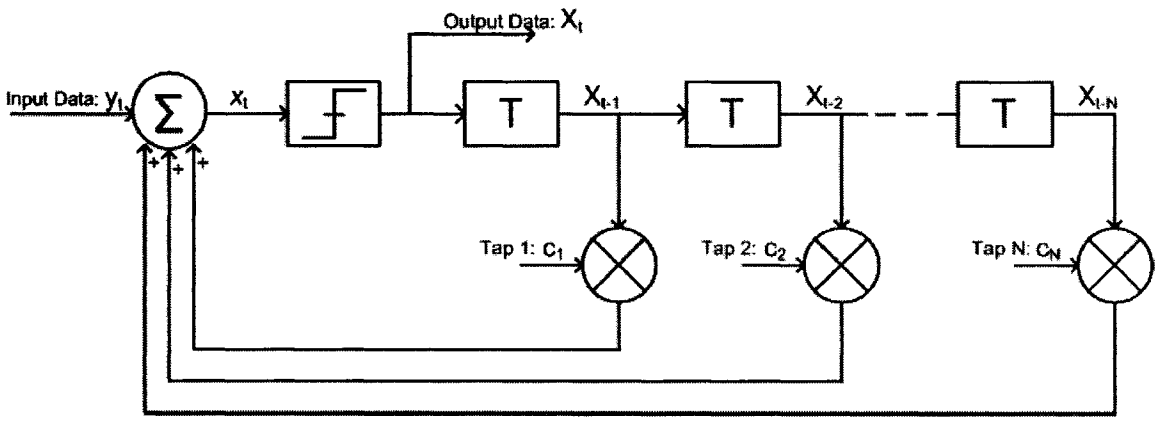

Figure 2.7 The Decision Feedback Equalizer

Through this decision on the incoming data symbols, the circuit is able to feedback a proportion of the previously received data bits without its associated noise or ISI. This allows the DFE to exactly cancel the post-cursor ISI from previous data symbols if

the decisions are correct and the tap values are chosen appropriately. The one consequence to this technique is that if incorrect decisions are made, the error will appear on the following data symbols because of the feedback loop in the filter. Although this will have a temporarily negative effect on the decision making ability of the filter, these errors are not considered fatal to its operation [7]. In order to fully understand the operation of this filter structure, one must look at the equation which describes its operation. Equation 2.4 below, describes the operation of the filter with respect to the input of the decision device.

$$x_t = y_t - \sum_{i=1}^{N} X_{t-i} \cdot c_{t,i} \tag{2.4}$$

The output of the summation node (x_t) is equal to the input from the channel (y_t) and the combination of the past decisions (X_{t-i}) and their respective tap coefficients ($c_{t,i}$) [12]. The coefficient term includes a dependence on time since it may vary according to an adaptation scheme such as the one proposed in this work. From the above relation, it is clear how the DFE can be effective in counteracting post-cursor ISI. However, it is also clear how important the selection of the tap coefficients is to the correct operation of the filter. The number of past decisions and coefficient values (N) used to remove the post-cursor interference depends on the length of the filter. The length is determined by the amount of ISI contribution from previous data symbols and the desired hardware complexity as there is typically more that one serial link in the system. Although the measur-

able effects from a transmitted data symbol can reach many baud-periods into future data symbols, as can be observed in the channel impulse response, the length of a typical DFE is at or under 5 taps [5], [12]-[15], [41].

The following section will discuss the various options when selecting the proper tap coefficient values. This is of paramount importance since these coefficients govern the performance of the equalization scheme as well as being of primary interest to this work. Following this, there will be a short overview of the current state-of-the-art implementation of a typical equalization scheme for a modern serial link. Specifically, the remainder of this document will deal exclusively with the decision feedback equalizer structure, since it has been shown to be both effective and popular amongst equalization techniques.

2.5 Adaptive Equalization

There are two separate methods for setting the tap coefficient values for any equalization filter. The first and simplest method involves setting the coefficients to an optimal value which is either fixed into the design or set at start-up based on the individual measured channel characteristics [5]. This is both costly and difficult and is therefore less desirable since each implementation would require measurement and adjustment from the designer [14]. Furthermore, a fixed equalization solution does not take into account the time varying nature of the channel characteristics [10]. These include temperature, humidity, power supply drift and component aging which can all change with time. Therefore an alternative solution which would make the coefficient values adaptive over time would be better suited to the changing environment of the serial backplane channel.

Adaptive equalization requires additional hardware complexity in order to evaluate the channel characteristics and make the necessary adjustments to the filter coefficients. The designer must select an algorithm which is able to quickly adapt to the channel and follow any changes over time with a reasonable degree of accuracy without excessive hardware and power expenditure. It is therefore important to carefully select an appropriate adaptive algorithm which is both effective in removing ISI as well as efficient in its use of valuable silicon area. The following section will review the most common algorithms used in the receive-end adaptive equalizer, and discuss the various trade-offs in their implementation.

2.5.1 Algorithms

There are many different algorithms which can be chosen to adapt the tap coefficients of the DFE structure. However, as mentioned earlier the designer must weigh performance over the cost of implementation. Therefore designers have typically selected the Least-Mean-Squares or LMS algorithm for use in the DFE since it is relatively straightforward in its implementation [9], [10], [16], [17]. Although there are alternatives such as the Recursive-Least-Squares algorithm which can perform much faster, it is however more complex and therefore more costly to implement [17].

The LMS algorithm acts to minimize the mean squared error between the output of the filter and the correct or ideal output. In order to accomplish this the algorithm adapts each coefficient based on the value and error of the data it is currently using. For each iteration, the algorithm computes the new coefficient value ($C(n+1)$) based on the previous value ($C(n)$) and a correction. The correction term is based on the value of the data ($d(n)$),

the error observed on that data (*e(n)*), and the correction or dampening factor (μ). This correction factor is one of the most important factors in determining the performance and stability of the LMS algorithm [16]. Unfortunately, the selection of the ideal convergence factor involves in depth stochastic analysis well beyond the scope of this thesis. For this reason, the convergence factor is typically set through the use of computer simulations [17]. Equation 2.5 below illustrates the operation of the full LMS algorithm.

$$C(n+1) = (C(n) + 2\mu e(n)d(n)) \tag{2.5}$$

The main deficiency of the LMS algorithm is that it requires the computation of the error, the multiplication with the dampening factor and data value, then the addition to the previous tap value. This may place excessive demands on the design performance, since it is expected to operate in a high-speed serial communications environment.

For this reason designers often select LMS variants which sacrifice some performance for a reduction in the design complexity and a timing constraints [17]. The following equations show the various alternatives to the full LMS algorithm [18].

$$C(n+1) = C(n) + 2\mu \,\mathrm{sgn}[e(n)]d(n) \tag{2.6}$$

$$C(n+1) = C(n) + 2\mu e(n)\,\mathrm{sgn}[d(n)] \tag{2.7}$$

$$C(n+1) = C(n) + 2\mu \,\mathrm{sgn}[e(n)]\,\mathrm{sgn}[d(n)] \tag{2.8}$$

In the above simplified versions of the LMS algorithm, the signum (*sgn*) or sign function is used to avoid numerous multiplications. Instead of using the complete value, only the polarity is used as a substitute which allows the use of bit-wise operations in

place of the full word multiplication [17]. These variants are named according to where this simplification is applied; equation 2.6 represents the sign-error LMS, equation 2.7 the sign-data LMS and equation 2.8 is the sign-sign LMS. Another important trade-off that must be recognized is that this simplification also sacrifices the final accuracy of the algorithm. The accuracy, otherwise called the coefficients residual error will not converge to zero due to the above simplification. Since slight errors will be given the same weight as larger errors, the algorithm will continually update the tap coefficients regardless of how small the errors become [17].

The other main performance sacrifice made in these simplifications, most prevalent in the Sign-Sign algorithm, is the slower speed of convergence. Since the tap coefficient updates will no longer be proportional to the magnitude of either the data or the error, they cannot make updates of variable size. This fact becomes clear in a binary implementation as each iteration can only alter the tap coefficient by the same amount each time. This lack of proportionality can create a condition of instability since random errors can cause incorrect tap updates, resulting in a temporary increase in error [17].

Finally, another trade-off can be made in order to reduce the timing constraints on the adaptation engine. This involves an alternative adaptation method which examines blocks of data then performs a single tap update based on the cumulative result at the end of each block [5], [14], [18]. This method is not to be confused with block processing, which adapts and filters blocks of data at a time. The block adaptation method does not alter the method of filtering, only the method of adaptation. Although this method will reduce the speed of adaptation significantly, it will also reduce the demands placed on the speed of the adaptive hardware [14]. Fortunately, this method also has an important bene-

fit in that it can also be used to stabilize the behavior of the adaptation. It accomplishes this by using the integrator structures typically present in the sign-sign implementation to reduce the unwanted variations in tap values [19], [20].

2.5.2 Training

In order to properly function, the coefficient values used in the filter structure must be correct. Ideally, the system would be given the correct coefficients during the design phase or be calculated at startup. However as discussed earlier, this would preclude the desired adaptive solution. Therefore the system must be designed to converge automatically to a set of coefficient values which allow for relatively error free data reception. To accomplish this, many systems use what is called a training sequence [3], [14], [21], [22]. This training sequence is a pattern of data transmitted across the backplane and is already known to the receiver. Typically, it is a pseudo-random bit sequence which contains a sufficiently wide spectrum to fully characterize the channel [7]. The receive-end equalizer is able to compare the received training data and use the local copy to generate an exact error value. Exact error values allow the adaptation algorithm to converge quickly without the risk of instability or mistake. Unfortunately, the use of a training sequence requires the storage (or generation) of the sequence in both the transmitter and receiver [21]. This has two major consequences, the first is the obvious increase in size of the transceiver architectures and more importantly cost [22]. The second is the fact that both the transmitter and receiver must share this common sequence exactly. This presents a difficult problem since the designer is now limited to mixing only compatible daughter cards, severely hindering design options.

Aside from the physical costs and design complexity, the use of a training sequence can consume a significant amount of time. In [14], the algorithm takes approximately 25 μs to converge the filter coefficients. At a data rate of 8 Gbps, this represents approximately 200,000 baud periods without any useful data transmission. Another proposal [23] has recommended an implementation which would sacrifice as much as 10% of the total bandwidth to periodically reset the tap coefficient values in order to reduce errors. Although the latter example is somewhat drastic, it highlights the dependence many topologies have on training sequences in order to ensure the performance of the equalizer structure. After this lengthy training period, the tap coefficients can either be fixed to this optimum solution or be allowed to continue adapting. Typically, the adaptive algorithm continues to operate by entering what is called the decision-directed mode. This mode allows the tap coefficients to be adapted based on the decisions made within the DFE, thus taking into account the long term variations in the channel characteristics [7]. Utilizing the error information from the received data, the algorithm is able to continually adapt without re-training. With such a technique applied to the startup sequence, several key improvements could be made to the equalizer structure, most of which have been noted in the above chapter. For these reasons, the primary goal of this work is to provide a new method for adapting the DFE without the use of a start-up training sequence.

2.6 State of the Art

Before proceeding with the description of the proposed solution, one must first look into a modern implementation of a serial link equalizer. This will allow a full understanding of the requirements which must be met by an adaptive algorithm, and how such

an algorithm will interact with the filter hardware. Furthermore, it will show the context in which the DFE filter will operate within the overall receiver structure. The implementations discussed here will be restricted to those which use standard CMOS and which contain a channel approximate to 34 inches of FR-4. The reason behind this distinction is clarity, as the available channel model is for a 34 inch FR-4 backplane and the proposed solution would be built using standard CMOS technology.

Many recent implementations of adaptive DFE structures have been published with link speeds of 10 Gbps and higher [41]. At these data rates hardware complexity has risen drastically. As a result, the receiver has several key components which directly effect the operation of the DFE, these are illustrated in Figure 2.8 below.

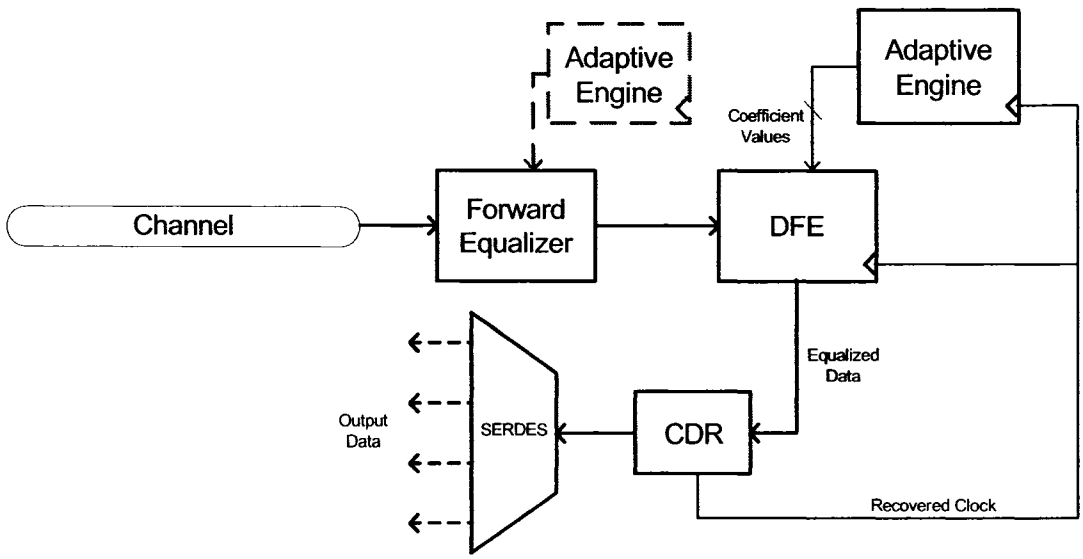

Figure 2.8 The Serial Backplane Receiver Structure.

The various components seen above are included in most modern implementations of the receiver topology. There are many different variations to this topology which may include additional blocks such as specialized amplifiers or input buffer structures [24],

[25]. To include all of such components would not be either appropriate or feasible in this document. Therefore, the first item of interest in the receiver topology is the forward equalizer since it is typically included. As mentioned earlier in this chapter, the forward equalizer structure can be moved to the transmitter in the form of a pre-emphasis filter. It should also be mentioned here that the forward equalizer need not necessarily take the form of a traditional FIR filter. As an alternative, a passive analog equalizer consisting of an L-R-C network can be used in order to reverse some of the frequency response of the channel [26], [27]. For this reason, in figure 2.8 the adaptation engine for the forward equalizer has been shown in grey as it may not necessarily be present. Following the forward equalization structure is the DFE. As discussed earlier in this chapter the DFE is composed of a tapped delay line filter, although its implementation was not described. Therefore, a brief description of the DFE architecture will be provided here.

Due to the timing requirements of the incoming data, the DFE filter must make use of high-speed circuitry and alternative configurations in order to operate effectively. These alternative configurations may include; half-rate topologies which reduce the timing constraints on each hardware block [27], [41], predictive first-tap feedback [25], [41], a sense-amplifier based decision device [15], [24], and most importantly, common mode logic [24], [25], [27]. Common mode logic or CML is commonly used in high-speed circuits due to the lower power consumption it exhibits compared to standard CMOS at high GHz Frequencies [28]. The lower power consumption is due to the fact that it uses a lower voltage swing and a constant current which is switched within the gate. Further benefits from the CML structure are derived from its differential topology such as common mode noise

immunity and compatibility with backplane signalling [29]. These benefits make CML type structures ideal for use in implementing the DFE filter. Figure 2.9 illustrates a typical modern filter structure which is similar to those used in [15], [24]-[25], [27].

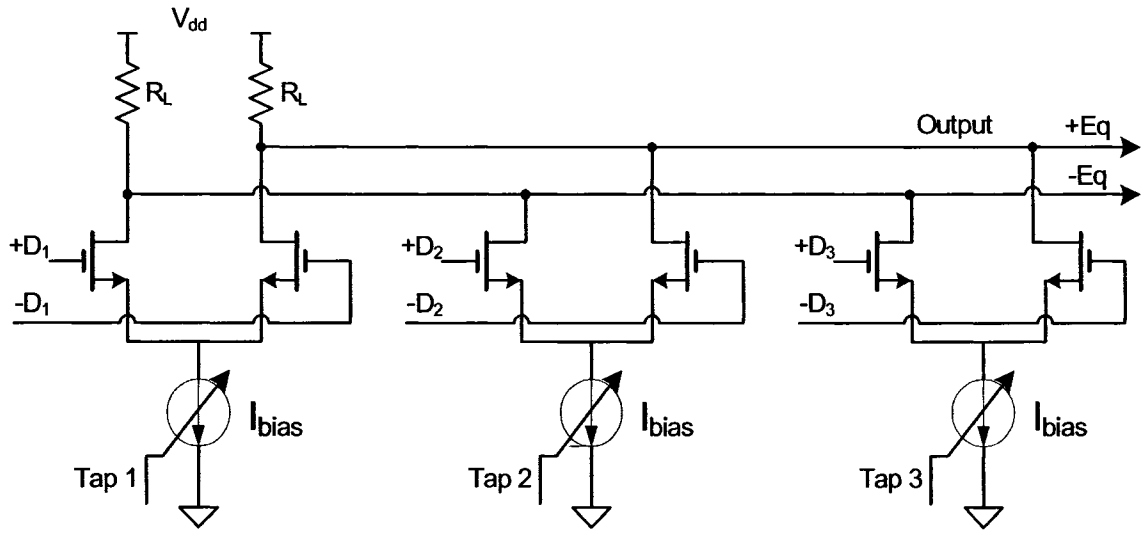

Figure 2.9 The Example CML Type Filter Structure.

In the above example, the present data is on input D_1 while previous data is fed back from the decision device and delay elements (not shown) as the inputs D_2 and D_3. In order to give each data bit a different weight, the bias current of each section is based on its associated tap coefficient value. Therefore the multiplication takes place in the analog domain. The bias current can be derived from a digital coefficient value with the use of a digital to analog converter (DAC) to give a bias voltage to the gate of a transistor [15], [42]. The precision of this digital storage is determined primarily by the desired accuracy and the capability of the design. The digital coefficient storage size in this work was set to 7-bits. Although it could be larger (as in [42]) it was set to be several bits larger than that of [5], in order to improve the stability of the system.

Due to the high-speed nature of serial backplane communications, it is not feasible or cost effective to transmit a synchronous clock signal on a parallel trace between transceivers as the delays seen through both paths would have to be made exactly identical [1]. Therefore, it is necessary to recover the clock signal from the incoming serial data and redistribute it locally in the receiver. The CDR operation is critical to all aspects of the receiver architecture. This includes the DFE filter and adaptation algorithm since both rely on operations which are synchronous to the incoming data. In order to accomplish this task, the CDR structure typically contains a local frequency synthesizer which is then locked to the incoming data frequency. This synthesizer is very similar to those used in other applications such as wireless communication devices and can be based on phase-locked-loop (PLL) or delay-locked-loop (DLL) type architectures or even a combination of the two [30]. The key component in the CDR structure however is not the frequency source, instead it is the phase acquisition and tracking system. This system must select the proper clock phase which will yield the most reliable sampling point within the incoming data. There are many techniques used to accomplish this many of which can be very complex [31], however the discussion of each is well beyond the scope of this thesis. It should be noted here that the type of CDR technique used can drastically affect the performance of the DFE adaptation. Due to the various signaling standards which can be used, there may be a requirement for the transceiver to able to operate at several different data rates. As a result, the time needed to acquire the incoming data frequency and phase may vary significantly between implementations. This is important since proper adaptation of the DFE coefficients cannot occur until the CDR has acquired the clock signal.

Finally, the next most important block in the serial data receiver structure is the deserializer, also commonly known as the serializer-deserializer (SerDes). It is the responsibility of the deserializer structure to convert the high-speed serial data stream into reduced speed parallel data words for use on-chip or on-board. Implemented using CML circuitry as used in earlier blocks, the deserializer is composed of smaller cascaded demultiplexer circuits in a tree topology [32]. The output of this block represents the end of the high speed data path and any following processing can therefore be implemented in standard CMOS circuitry.

TABLE 2.1: Summary of components found in a State-of-the-Art Equalization scheme.

Reference #	Component	Technology	Data Rate & Signalling	Notable Feature
11	FFE and Crosstalk Cancellation	0.18um CMOS	20Gb/s 4-PAM	New tap multiplier circuitry
20	Adaptation scheme	1um CMOS	150Mb/s (Disk drive)	Block Adaptation technique
19	Adaptation scheme	0.5um CMOS	100Mb/s (Disk drive)	Pre-counter to eliminate tap noise
24	Bit Detection Scheme	0.13um CMOS	6.25Gb/s Binary	Sense-Amp based detection
15	Adaptation scheme	0.13um CMOS	6.35Gb/s Binary	Sign-Sign LMS
27	Half-Rate	0.18um CMOS	10Gb/s PAM-2	Speed and architecture
41	DFE-CDR Relationship	90nm CMOS	10Gb/s PAM-2	CDR totally independant of DFE structure

2.7 Summary

It has been the goal of this chapter to provide the reader with and overview of the issues surrounding high-speed serial backplane communications. These issues include; severe ISI, attenuation, crosstalk, and clock recovery. In order to address these issues, a variety of circuit techniques were presented with an emphasis on the adaptive decision-feedback equalizer. To operate effectively, this equalizer must rely on the adaptation algorithm to select the filter coefficients which allow for the recovery of the clock and data. Unfortunately, many implementations of the DFE adaptation engine rely heavily on specialized training sequences to accomplish this task. These training sequences offer reliable performance at a significant cost of area, within the adaptation engine, and time which could be used for transmitting data. It is therefore the goal of this thesis to develop an alternative method of adapting the DFE coefficients without the use of such a training sequence.

CHAPTER 3

Blind Adaptive Equalization

3.1 Introduction

This chapter will present a model design of an adaptive engine which can converge the coefficients of a 4-tap DFE to acceptable levels without the use of a training sequence. Ideally, this engine would be implemented and tested using an existing serial backplane in order to fully verify its operation. This would involve implementing a full serial transceiver, which would be beyond the scope of this work. Therefore it would not be feasible to attempt such an implementation in order to prove the operation of this blind adaptive equalization method. As an alternative, the proposed engine will be proven through the design of a Matlab based model. This, in combination with the accurate measurement based model of the serial channel shown earlier, will ensure the validity of future Matlab simulations. While these simulations will be reserved for a later chapter, the current chapter will focus on the decisions made in the design of the adaptation engine. These decisions include; the algorithm used, the method of error detection used, and the various topology simplifications which have been found necessary for operation at the desired data rates.

3.2 Algorithm Design

In order to simplify the design process of the blind adaptive equalizer, it was decided that the method of adaptation will be based upon an existing algorithm. Rather than attempt a completely novel solution which would require a strict theoretical analysis, a well characterized algorithm will be used to provide an already proven foundation. Therefore the first and most important decision to be made is the selection of the standard algorithm on which the proposed solution will be based. As discussed in the previous chapter there are several algorithms which can be used for this application, each with varying degrees of complexity. Since the application for this method will be in a high-speed binary signalling environment, an algorithm with a reduced complexity would be most appropriate. Therefore, the sign-sign LMS, which is one of the most simplified LMS algorithm variant has been chosen for this purpose. The sign-sign LMS, as discussed in chapter 2 and repeated here in equation 3.1, utilizes only the sign of the data and the sign of the error for adaptation.

$$C(n+1) = C(n) + 2\mu \operatorname{sgn}[e(n)] \operatorname{sgn}[d(n)] \qquad (3.1)$$

This simplified algorithm is only the starting point, with the remaining challenge being the extraction of reliable data and error information from the incoming symbols. Due to the severe amount of ISI present in the received data, it is unfeasible to simply run the above algorithm in what is known as decision-directed mode. This would involve comparison of the incoming symbols to their ideal values, and would also rely too heavily on the individual decisions made on the data values. The result would be tap coefficients not converging to acceptable values, but rather drifting uncontrollably. Therefore, the

incoming symbols must be dealt with in a novel method which can allow acceptable convergence while at the same time reducing the amount of incorrect decisions and their effects. Furthermore, the information that is gathered within this novel method must be processed in a manner which is feasible at the desired speeds. This requirement implies that methods such as over-sampling, which involves information gathered beyond the Nyquist sampling rate, would not be advisable due to excess power and area demands.

The following sections will be devoted to discussing the overall solution and to illustrating how this proposed solution addresses each of the above concerns. Beginning with a description of the model, the following will include an overview of each of the proposed hardware blocks which directly enable blind adaptation of the DFE.

3.2.1 Structure

The proposed solution of the blind adaptation problem can be summarized in several key areas. Specifically the areas of; how reliable error information is acquired, how to reduce excess variation on the tap coefficients and how to allow the algorithm to operate at high data rates will be discussed. In order to understand how each of these issues will be addressed in this work and how they interact with each other, one must look at the overall structure of the proposed solution. Figure 3.1 on the following page, illustrates the simplified topology of the typical DFE implementation. Starting with the error detection method, information is then passed to the algorithm hardware where specialized circuitry is used to process the incoming data and error information. After the algorithm is applied to the incoming information, the tap coefficient values are updated.

This adaptation process is occurring continually and in parallel with the filtering of the incoming data. It should also be noted that the algorithm as well as the filter use the same decisions which are made within the DFE structure itself.

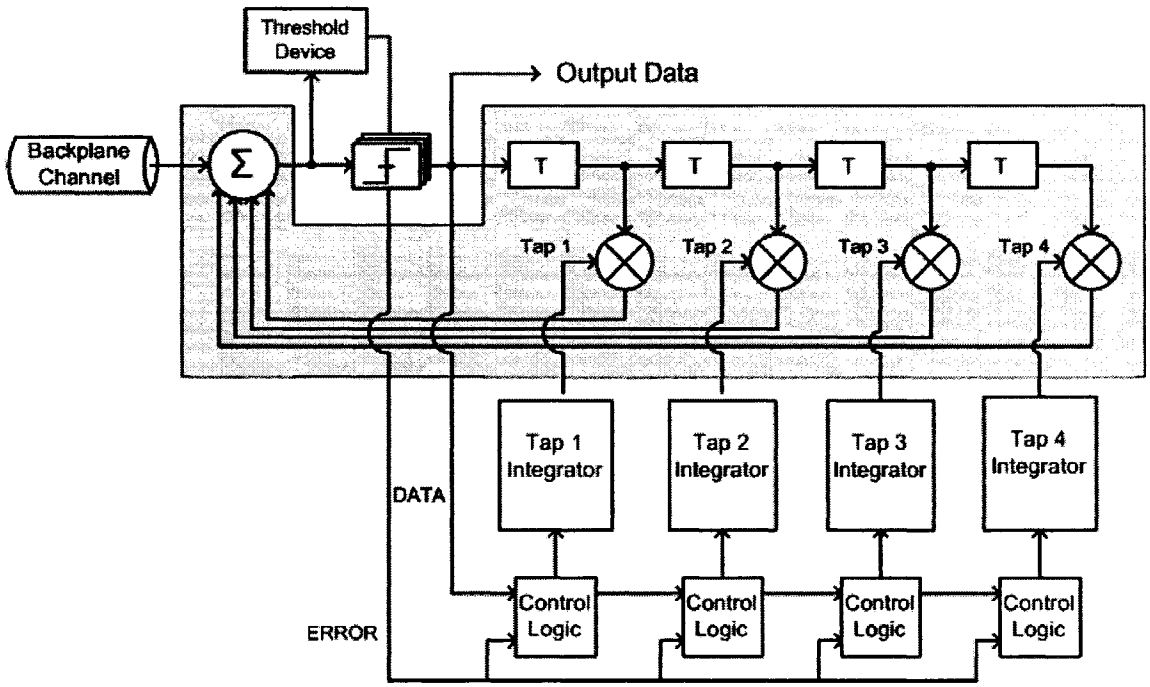

Figure 3.1 The Simplified Model Topology with Emphasis on the Adaptation Structure.

3.2.2 Error Detection Method

Without a training sequence, the adaptation algorithm no longer has access to a known sequence from which exact error information can be generated. Therefore the error information used must be based on decisions made locally within the receiver. As stated earlier, the sign-sign LMS algorithm is typically unable to converge the tap coefficients to an appropriate level while using the traditional decision-directed mode. This traditional mode would involve making a decision on the value of the incoming data, then computing the error based on the difference between what was received and the ideal signal level. In

the NRZ signalling environment used for this work, the ideal value for a binary 1 is +0.5V, and for a binary 0, -0.5V. However, due to the amplitude attenuation and the ISI across the channel, the received amplitudes are rarely at the ideal levels. Furthermore, due to the sign function being applied to the computed error, the magnitude of the error is deleted. Therefore both slight and major amplitude errors are treated equally within the adaptation algorithm. This results in an excessive amount of error information being generated which leads to an excess amount of tap variation. It is this severe tap variation which prevents convergence. To combat the tap variation issue, a new approach was developed to issue more meaningful error information while still relying on the simplicity of the sign-sign algorithm. In this new approach, rather than making a comparison between the received signal and the ideal value, the incoming data will be compared to a level which can scale with the overall amplitude obtained at the receiver. This level will form a metric in order to gauge the reliability of the incoming data symbols. This requires the use of three separate bit slicers at the front of the receiver.

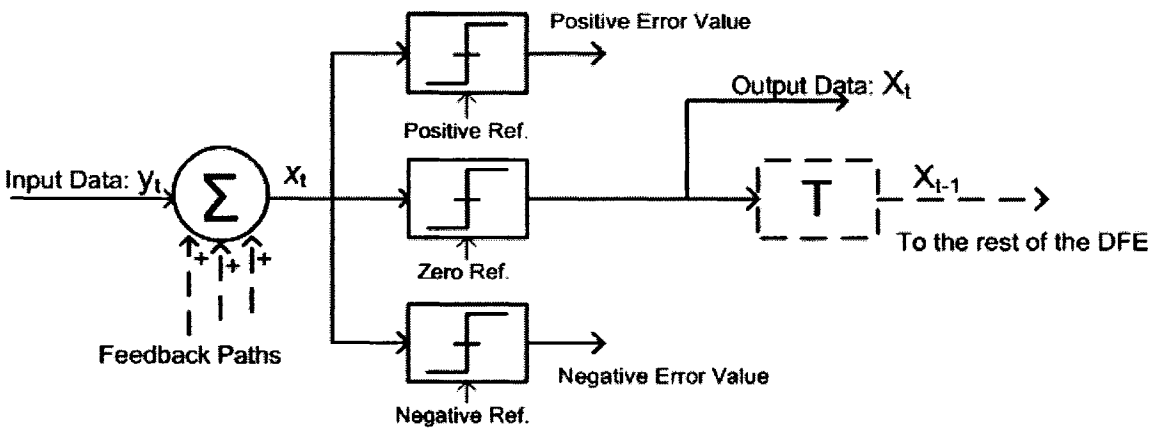

Figure 3.2 Bit Slicers within the DFE Structure.

The main (zero reference) slicer will make decisions on the value of the data symbol and will be centred at the zero level as in the typical implementation. The two other slicers will use separate reference levels which are between zero and their respective maximum amplitudes. These three slicers will be within the DFE structure, between the summation node and the rest of the filter as illustrated above in figure 3.2.

The reference levels given to the error slicers will be set at a value which will allow for the removal of the bulk of unreliable update information. If an incoming symbol is greater than a certain percentage of the average amplitude, it is assumed to be correct, and the error signal will remain zero. The exact value of this threshold will determined empirically through computer simulations, which will be discussed in further detail in the following chapter. This new criteria results in smaller amplitude discrepancies being ignored instead of being given equal weighting as in the previous algorithm implementations.

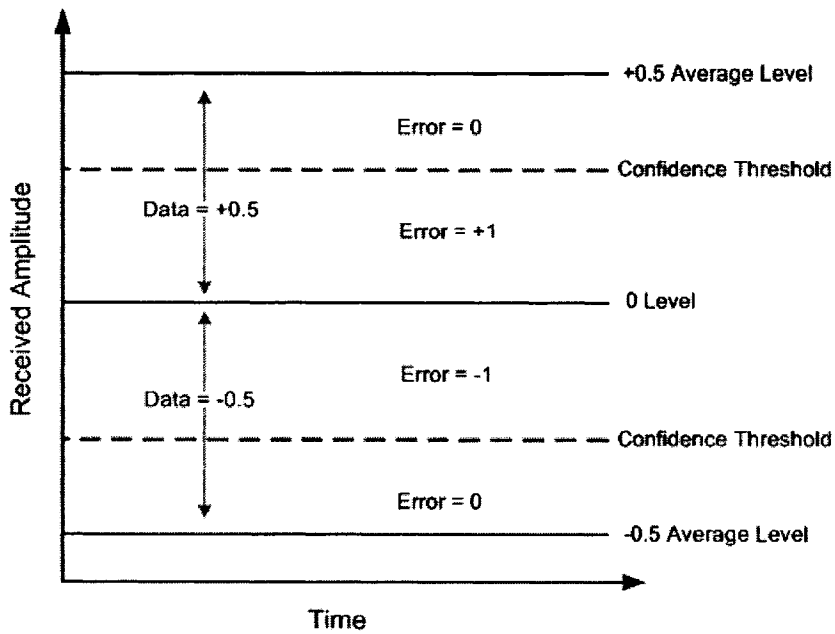

Figure 3.3 Data and Error Decisions Based on Received Amplitude at Slicers

This will reduce the total number of error signals given to the algorithm, and will help prevent erroneous updating of the tap coefficients. It is important to note that this will impact the residual error on the tap coefficients after they have converged, however without ignoring the small amplitude errors the algorithm would be unable to converge. Furthermore, by using only larger amplitude discrepancies the algorithm will be focused on correcting the more severe ISI while ignoring the minor distortion from other sources. Figure 3.3 above, illustrates the relation between the incoming amplitude from the channel and the error bit given to the algorithm.

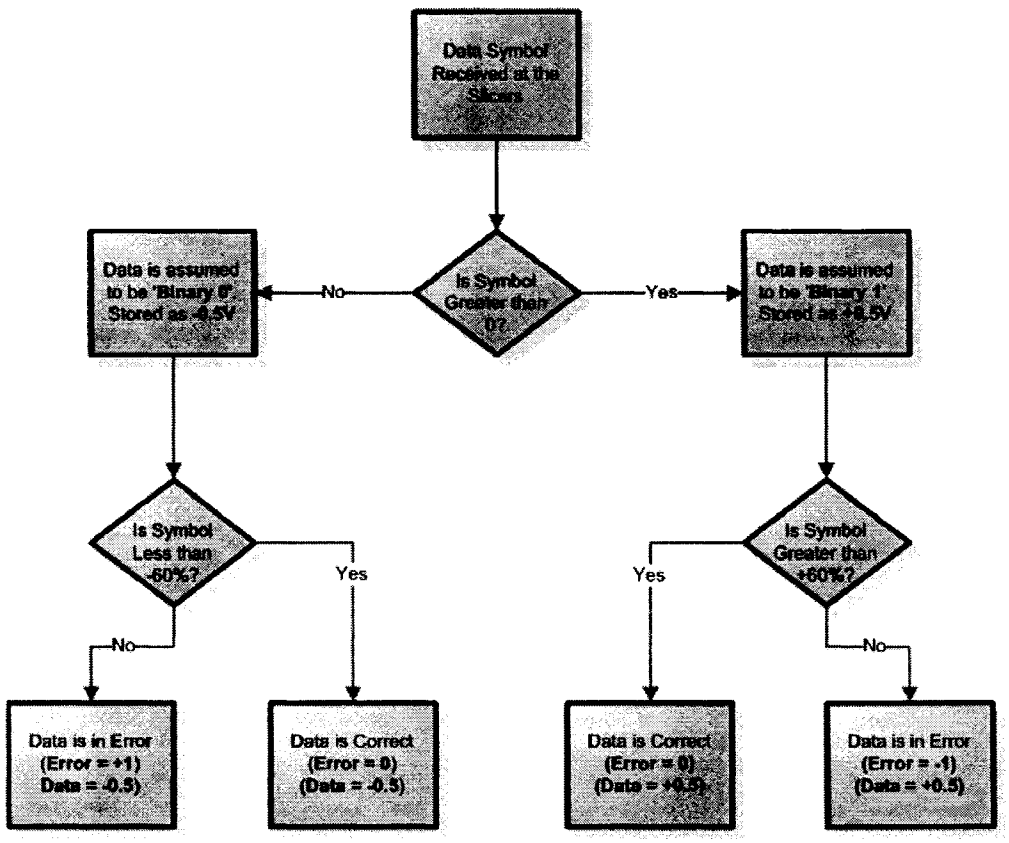

Figure 3.4 The Decision Flowchart of the Input Slicers

Figure 3.4 above, illustrates the 4 possible outcomes from the input slicers, and how each of the decisions are made based on the criteria shown in figure 3.3. It is important to note that the two center outcomes assume that the received data is entirely correct.

This is reflected in the fact that the error bit is set to zero in both cases. As a result, they will not influence a change in the tap coefficients. Conversely the two outcomes on the periphery will issue error signals which will influence a change in the tap coefficients. It is this zero-error result which allows the algorithm to exclude small amplitude discrepancies and only use large amplitude errors to move the tap coefficients. Furthermore, the error as well as the data bits will maintain polarity information for use within the algorithm.

In addition to the added slicers, there will be a need for extra circuitry to provide the required reference levels. This reference circuitry must be able to quickly acquire the incoming amplitude at startup in order to allow the algorithm to begin convergence as soon as possible. The reference generator must also track any changes over time, as to preserve the adaptive equalizer's ability to adjust to long term changes in the channel. These issues will be further addressed in the proposed architecture shown in chapter 5.

3.2.3 Block Adaptation

After the incoming symbols have been combined with the feedback from the remainder of the DFE structure and have passed through the input slicers, two information bits will have been generated. The first is the data value and the second is the error value. Unfortunately these output bits will be at the same rate as the incoming symbols, making them difficult to process. It would therefore be difficult to update the tap coefficients by directly using these bits. For this reason, a block adaptation topology was selected. This topology will compute an adaptation direction for each Baud period and accumulate the overall result from many data symbols using a counter. This accumulation acts as a low-

pass filter to reduce unwanted tap variations, before applying a correction to the coefficient [19]. The block size can be set to any value which would give a sufficiently reduced timing requirement for implementation of the algorithm. The block size used for the model designed in this work will be discussed in the following chapter. Figure 3.5, illustrates how the algorithm is applied to each set of data and error bits, with the result being stored in the counter.

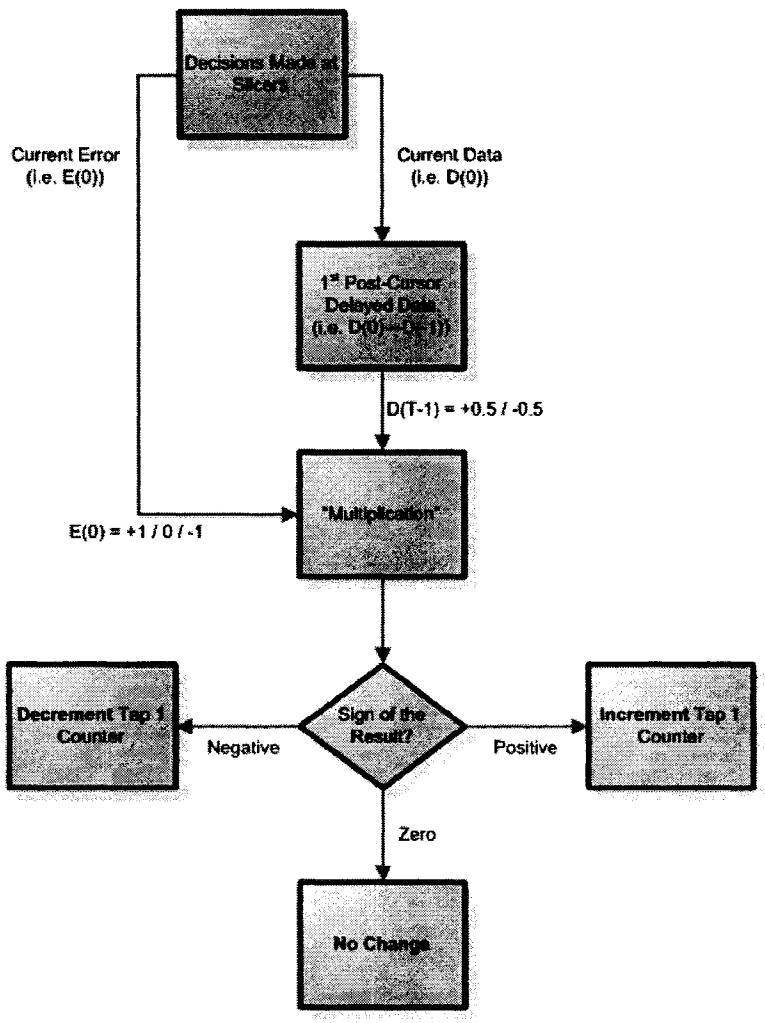

Figure 3.5 The Modified Sign-Sign Operation in Block Form for Tap 1

3.2.4 Tap-Noise Elimination Pre-Counter

While the above strategy will reduce the timing constraints on the tap update hardware, it will not necessarily prevent incorrect tap updates from occurring. Although the new method of error detection presented in the previous section will help to reduce the number of unnecessary tap updates, there will still be a significant amount of incorrect decisions being made at the input slicers. These incorrect decisions will occur on both the polarity of the data, as well as the error values.

Steps must therefore be taken in order to reduce the overall effects these incorrect decisions have on the ability of the taps to converge to an appropriate value. The first such step used in this model is a method of reducing the effect of each individual tap coefficient update. For the model proposed here, the tap updates are reduced to single steps in either the positive or negative direction. Throughout each block, the sign-sign algorithm is used to determine an update direction for each individual data-error pair. These updates will be accumulated over the length of the block using a counter for each tap as described in the above section. At the end of the block, a single update will be made to each coefficient by incrementing or decrementing each respective binary word. This can be easily implemented since the tap values will also be stored in the digital domain using counters. As a secondary measure, a threshold will be used to evaluate whether or not a tap coefficient update is warranted. If the number of accumulated tap updates in one direction is above this threshold, then the coefficient is updated by one step in that direction. If the accumulated updates are below this threshold, the tap coefficient is left unchanged and the update counter is reset before the next block begins. The result of having this threshold is a significant reduction in the amount of noise experienced by the tap coefficients during and after

convergence. At the same time, significant shifts in the coefficient values will be left unhindered since the sign-sign algorithm would accumulate a large value on the tap counter. Furthermore, the update speed will be largely unaffected since it is determined primarily by the size of the data blocks. Figure 3.6 on the following page illustrates the decision process made during each baud period, with tap updates only occurring at the end of each data block according to the criteria outlined above.

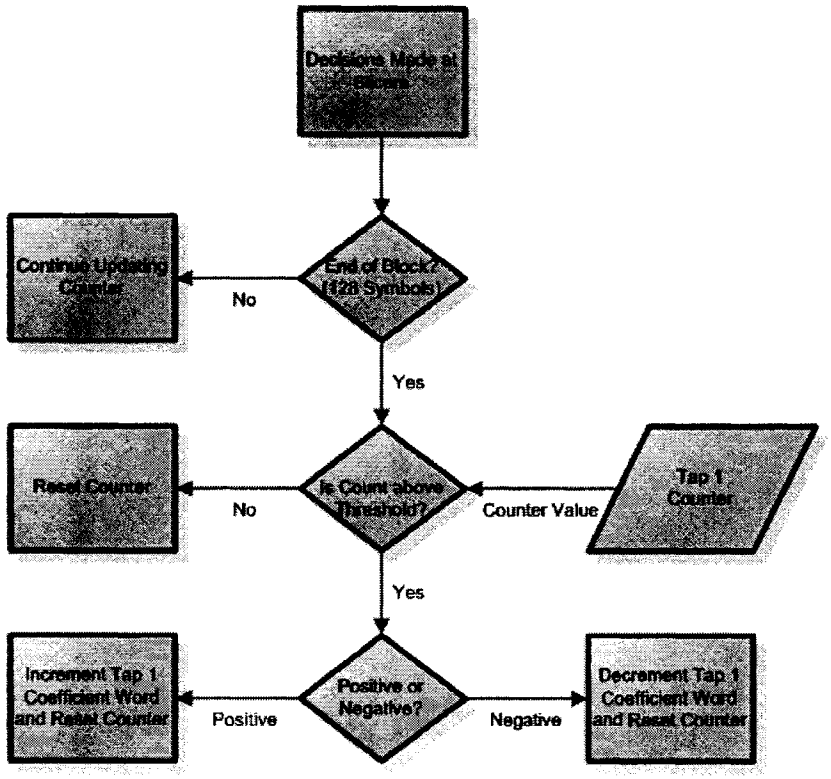

Figure 3.6 Coefficient Update Criteria for Tap 1.

The threshold level itself will be determined through computer simulations in Matlab, and is based on an optimum reduction in necessary tap variation while at the same time allowing the coefficient to change freely. It would therefore take a final counter value greater than this threshold in either the positive or negative direction at the end of the block to initiate a change in the tap coefficient.

It should also be noted that this threshold value is heavily related to the block size used in the adaptation, therefore larger block sizes would require larger threshold values.

3.2.5 Final Algorithm Topology

With all of the individual aspects of the blind adaptation method now fully explored, one must look at the interaction between these individual sections and the overall DFE structure. Figure 3.7 below shows the overall structure of the DFE filter and algorithm hardware. The shaded area contains an idealized DFE filter structure, although it is present in the Matlab model, it is not the focus of this work. Therefore, its exact implementation will not be discussed further in this work. Outside of the shaded area, after the summation node of the filter is the threshold circuitry and the added slicers for making data and error decisions. Below the filter lies the hardware necessary for the each tap adaptation which includes; two counters, one digital-to-analog converter (DAC), and control logic for implementing the sign-sign function.

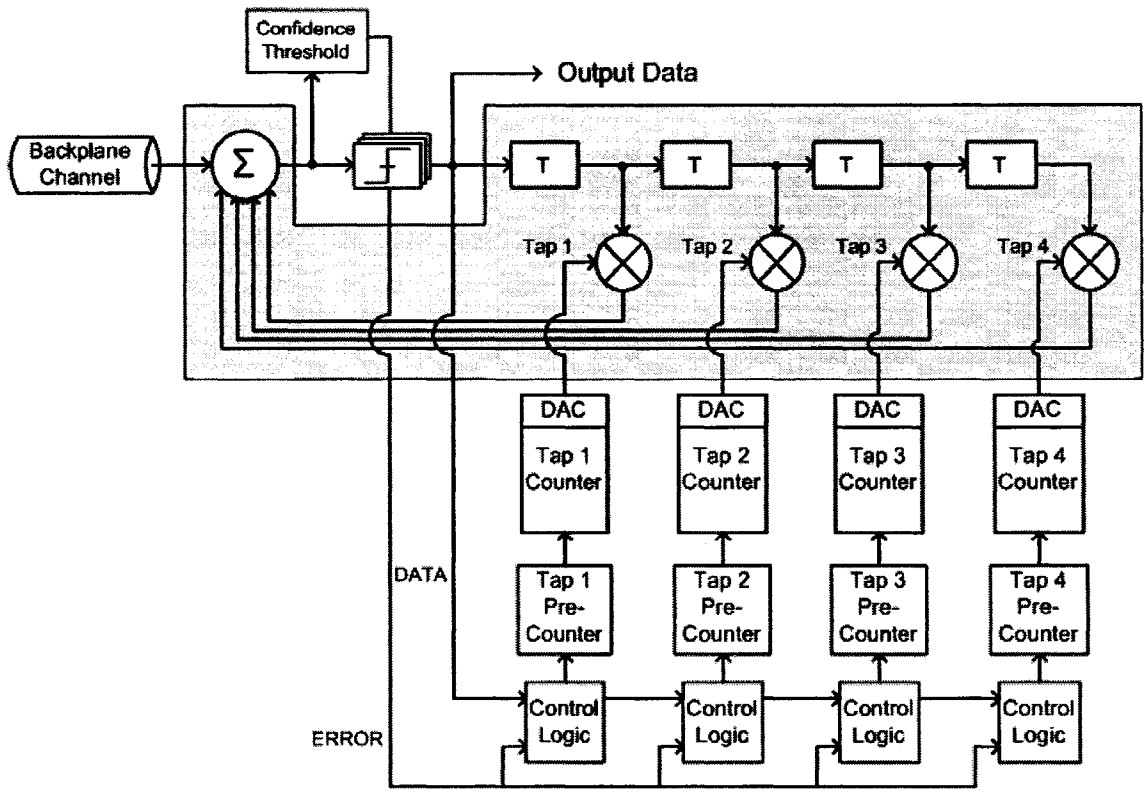

Figure 3.7 The Overall Topology of the Receiver with the Adaptation Hardware and DFE Filter (in grey).

The following section will be devoted to a detailed discussion of the model as created in Matlab. The details will include a discussion of the previous work, the exact model parameters, and the methods used to implement the above hardware blocks. This overview is necessary in order to fully understand the simulation results which will be presented in the following chapter.

3.3 The Matlab Model

The model design of the blind adaptive equalizer is based on a previous model by Lei Lin in [5]. In this previous work, an adaptive pre-emphasis filter was designed and built using a Matlab model involving the same channel as used in this work. Therefore several parts of the previous model have been reused, particularly the transmitter pulse shaping and the convolution of the data with the channel response. Also reused is the digital to analog converters for converting the digital tap coefficient words into analog multiplication values for use in the filter structure. These parts will be discussed further below as each section of the model is briefly reviewed.

3.3.1 Part 1: Initialization

The first section of the Matlab model is the initialization of the memory and variables that will be used. This initialization includes the settings for all of the hardware blocks including; the tap coefficient precision, the coefficient ranges, and tap counter sizes. To increase the speed of simulation and to hold memory usage to a reasonable level, all large data structures will be pre-allocated before use and cleared to zero afterwards.

One important set of parameters which directly governs the adaptation behavior is the tap precision and range. The tap range is set to allow any of the 4 coefficients to travel between 0 and 0.5, these values are with respect to the zeroth tap value of one. That is, the symbols from the channel entering the filter are considered to have a tap coefficient of one while the other taps are normalized to this value. The maximum value of 0.5 has been selected in order to maintain stability as it is unlikely that the post-cursor ISI will be greater than half of the transmitted symbol amplitude. Having a tap value greater than this

could lead to error propagation under some conditions [25]. Finally, the coefficient word size is set which will determine the precision of the tap values as well as the minimum step sizes. It should be noted here that the precision of the tap coefficients is highly dependant on the hardware implementation. For the purpose of this model, a coefficient word size of 7 bits was chosen. This allows for a sufficiently small step size for use in simulation, while at the same time it remains within a reasonable word size which could be fabricated.

The final variable to be initialized is the window size used for measuring the output magnitude. This magnitude will be used to generate the error decision threshold described earlier. Since it is intended that a simple analog circuit would be used for this purpose, it is necessary to replicate its behavior within the model. Therefore a windowing technique was used to provide a measurement based on the most recent past symbols in a manner similar to a filter. Although this method is very simplified, it allows the model to track the incoming amplitude over time throughout the adaptation process. The window size used for this model will be evaluated in the following chapter. It should be noted however that one must select a window size which will respond quickly to the incoming signal but it should not interfere with the overall performance of the rest of the adaptation engine.

Following the initialization, extraction of the channel impulse response is done using the measurement data from the IEEE standards committee [40], [42]. This channel impulse response is used to shape the data and re-create the ISI conditions found in the actual backplane environment. The extraction, as well as the pulse shaping, is done in the exact same manner as the previous work [5].

3.3.2 Part 2: Transmission

The next step in the Matlab model is the synthesis of the data sequence for transmission across the channel. For this model, two different types of data streams were used; the first uses the built in Matlab random number generator, while the second uses a pseudo-random bit sequence (PRBS) generator. The built-in sequence offers a bit stream created from a random number being applied to a binary threshold, without a significant amount of effort.

The PRBS generator is used to model the typical methods used in testing serial transmission systems. Since it is not possible to create a small circuit that can reliably produce random data, PRBS generator circuits are often used which contain a simple shift-register and feedback paths [4]. For this model several PRBS sequences will be used, including shorter sequences like 7, 14, and longer sequences like 63. The longer sequences provide strings with a greater number of consecutive ones or zeros which contribute severe ISI. This makes them ideal for testing serial transmission circuitry as they would expose any weaknesses in the system.

After the data has been generated, it is shaped according to the channel impulse response extracted earlier. In addition to the shaping, a limited amount of noise is also added. This noise, which also appears in the previous work, helps to model the effects of crosstalk from adjacent channels as well as other random fluctuations. These fluctuations can be attributed to power supply spikes or other unrelated environmental sources.

3.3.3 Part 3: The Receiver

The first part of the receiver section of code involves setting up the model in order to process the incoming symbols. First the channel output is down-sampled and aligned so that it can be detected and processed correctly. Although this work does not include a CDR circuit model, this section performs the data alignment assuming ideal clock recovery. Further into the receiver section of the code, the filtering and adaptation is performed inside of a loop. At the beginning of this loop the delay line, which will form the DFE filter, is implemented with careful attention being paid to the start-up conditions. Next, the actual filtering is performed and the result at the summation node of the DFE is computed. This result is stored in the variable called 'temp_data', and will form the basis for all aspects of the adaptation process that follows. These aspects include the three main variables which will be used during adaptation. The first is the error threshold, called 'ErrThresh' and is based on the windowing technique which was discussed earlier in this chapter. The second is the output of the data slicer which is stored as a vector called 'deci_data', this output data is based on a fixed threshold at zero. Finally, the error threshold is used to determine the values of the error bit which corresponds to each data bit. These error bits are kept in the vector called 'Block_Errors'. With the values stored in these two vectors being computed and updated at each baud period, the adaptation process may proceed normally.

3.3.4 Part 4: The Adaptation Engine

The second part of the receiver and the focus of this work, is the adaptive engine. This section is split into two separate halves; the first implements the tap pre-counters and uses the data and error bits from part 4, while the second implements the tap coefficients

and their updates. In the first half, each of the past data bits and the current error bits are multiplied together at every bit period. The sign of the result is then used to determine an update direction for each of the tap pre-counters. The first tap pre-counter is stored in the variable 'Tap1cursor', and it is updated throughout the length of each block until the end where it is evaluated and reset. It is in this multiplication and storage where the Sign-Sign LMS algorithm foundation is used to adapt the tap coefficients.

With the tap pre-counters accumulating the results from the sign-multiplications, the results must now be passed to the second-half of the adaptation engine in order to update the tap coefficients. Unlike the first half, the second half of the adaptation engine is only enacted at the end of every data block. Once the end of the data block is reached, the final pre-counter value is compared to the threshold in both the positive and the negative direction. Furthermore, the current state of the tap value is also evaluated to see if it is at the maximum or minimum boundaries. Using the above criteria, any necessary tap updates are made by updating the individual digital storage words which would then be passed to the digital-to-analog converters. Regardless of the update direction, or whether or not the taps are updated, the tap pre-counters are reset to zero for the next block. After the tap updates are complete, the tap coefficients will remain fixed throughout the next block while the tap pre-counters discussed earlier will continue to be updated. In order to track any sudden changes in the signaling environment, the above adaptive engine will continue to operate after the tap coefficients have settled to a final value.

3.3.5 Part 5: Output

The final section of the Matlab model, has no purpose in the adaptation system itself, rather it is simply used to create the output diagrams which will allow critical examination of the model performance. The most important diagram which will be used in this purpose is called the eye-diagram.

Eye-diagrams simply overlay many, aligned baud-periods of the output and allows the viewer to observe the quality of the equalization. It is termed an eye-diagram because symbols that do not contain significant ISI or symbols which have been equalized appear to form open "eyes". The section of the eye with the widest vertical opening represents the ideal sampling point. If a decision is made at this point, the bit slicers have the greatest chance of correctly detecting the value of the data symbols. Unequalized or poorly equalized symbols appear to have closed "eyes", thus making it more difficult to properly detect the value of the incoming symbols. Therefore, the greater the eye-diagram appears to be open, the better the equalization and data reception. The eye-diagrams presented in the following chapter will be key in demonstrating the performance of the adaptive method presented here. In addition to eye-diagrams, graphs showing each of the tap values over time will be used to prove the performance of this system.

3.4 Summary

The above chapter has presented the full Matlab model and the concept behind the method of blind adaptation of the decision feedback equalizer. It has been the goal of this chapter to clearly describe the method of blind adaptation through three separate steps. The first was to describe the general technique which was developed, while the second

focused on the specific details of its operation. Finally, the details of the model were fully elaborated with specific focus being placed on how it was designed to replicate actual circuitry.

Although it is not feasible to design and build a full transceiver, the presented model utilizes measured channel data in order to test the concept within a backplane environment. Significant steps have been taken to ensure that the model replicates the behavior of digital circuitry. The concept of the blind adaptation strategy is centered around the novel method of generating error information, based on the incoming amplitude. This new method will allow smaller and insignificant amplitude errors to pass without affecting changes to the tap coefficients. With the error information simplified and reduced in volume, the model is able to make the necessary changes to the tap coefficients with a reduced amount of misadjustment. Through the design of the Matlab model, it is also shown that the adaptation method can operate at a reduced speed in order to better facilitate implementation. Furthermore, the block adaptation method allows for the reduction of tap noise through the use of the tap pre-counters.

In the following chapter, the Matlab model will be elaborated and tested using several different data sequences and data rates. It will also be analyzed for stability and vulnerability to variations in the key components outlined above.

CHAPTER 4

Simulation Results

4.1 Introduction

This chapter will illustrate the performance of the proposed adaptive engine under various conditions with the use of Matlab simulations and the channel measurement data. The primary goal of this section is to elaborate the specifics of the design as well as demonstrate the reliability of the proposed solution. Emphasis will be placed on the design choices discussed earlier in chapter 3, their effects on the overall performance as well as the various design alternatives available. The performance of the blind adaptation method will be compared with the trained version using the same hardware model. This allows the trained version to make changes to the tap values that are known to be correct while allowing for a useful comparison to the blind version. This comparison is accomplished by altering the same variables in both designs and comparing the performance differences between them. Although there are many variables in the design which can be altered over a wide range, only a select group of important factors are included here in order to maintain clarity. With this limited set of variables, it will be shown that the blind adaptation method performs at a level similar to that of the trained solution, therefore eliminating the need for the training sequence.

4.2 Simulation Setup

The simulation results presented here are divided into three separate categories. The first involves simulations of the proposed design as presented in the earlier chapter, with a depiction of the incoming and outgoing data. The second section, involves several different analyses involving each of the design variables and how they affect the performance of the adaptation engine. Finally in the third section, a brief study of how the engine will perform under the influence of noise within the filter structure. In all of the simulation results, the data rate used is 10 Gbps which is equal to a baud period of 100 ps. This is the highest data rate at which the presented method will reliably operate without any other forms of equalization such as transmitter pre-emphasis or feed-forward equalization.

In order to rigorously verify the performance and validity of the blind adaptation method, each of the simulation results shown, except where indicated, make use of normally distributed random data. This random data is generated with the built in 'randn' function of Matlab. Also, the state of the random number generator is set to a new value each time the model is run. As a result, there is no repetition of data patterns between simulations which could either hinder or aid the convergence of the adaptation engine and therefore skew the results.

Although eye-diagrams are typically used to evaluate the performance of a serial link, only the first section will use eye-diagrams to demonstrate performance. The second and third sets of simulation results will show only the tap values. However, as the first set will show both tap coefficients as well as the resultant eye-diagram, a direct link between coefficient values and eye-opening will be clear.

In all of the following simulations, the input to the DFE is the raw and un-equalized data. The eye-diagram in Figure 4.1 below, contains many successive baud periods of the received data symbols after passing through the channel model. Without any equalization the data symbols have been severely distorted due to the ISI introduced by the channel, and the transitions between the two data periods is totally obscured. Futhermore, there is a significant number of data symbols appearing near the zero crossing thus completely closing the eyes. From this diagram it is clear that the typical decision-directed method would have a very difficult time in adapting the DFE coefficients.

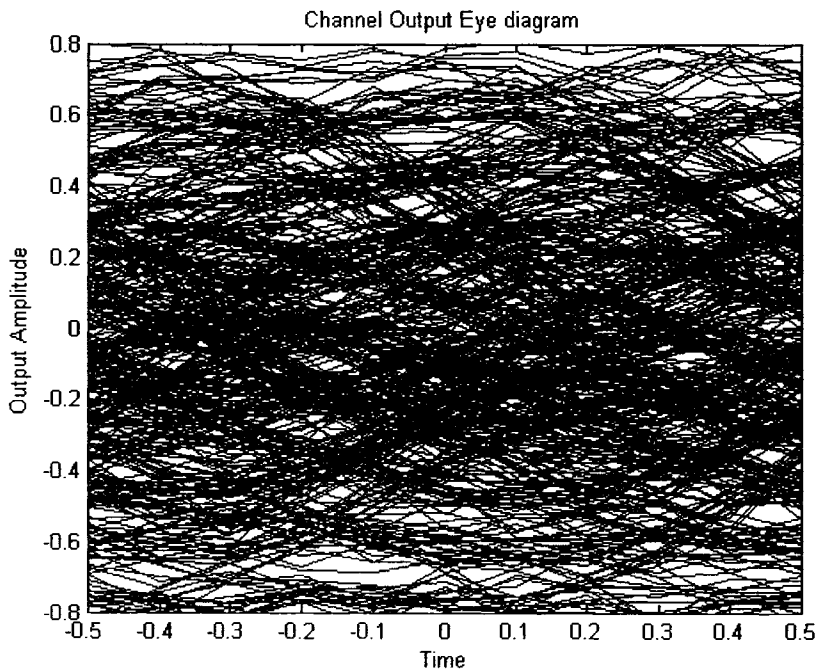

Figure 4.1 The Raw Channel Output (in Volts) at 10 Gbps.

With the above input, the following results will show the tap coefficients being converged from the initial starting condition. This starting condition sets the first and second tap coefficient words at 32b and 16b respectively, while the remaining taps three and

four are set to zero. This starting condition serves to reduce the overall convergence time and increase reliability of the adaptation engine. This startup condition will be discussed further in the following sections.

4.3 Simulation Results - Designed Conditions

The first set of simulation results will show a comparison between the blind adaptation and the trained adaptation under the optimized settings. These optimized settings were determined through extensive computer simulations which will be discussed in further detail in the following sections. The table 4.1 below, summarizes the properties of the optimized solution. It should be noted here that these settings do not represent a unique solution for blind adaptation, instead this solution gives the fast convergence with relatively little tap noise.

TABLE 4.1: **Parameters of Optimized Design**

Parameter	Value
Data Rate	10 Gbps
Block Size	128
Error Threshold	45%
Tap Update Threshold	8

The following figures 4.2 and 4.3 show a direct comparison between the convergence of the adaptation engine with and without the use of the training sequence. From these simulation results it is clear that the performance of the adaptation system is almost identical in both cases. Therefore the adaptation speed is not dependant on the use of a training sequence, rather it is dependant on the simplifications that have been made to the adaptation system. These simplifications include the use of a sign-based algorithm, block adaptation, and the error threshold.

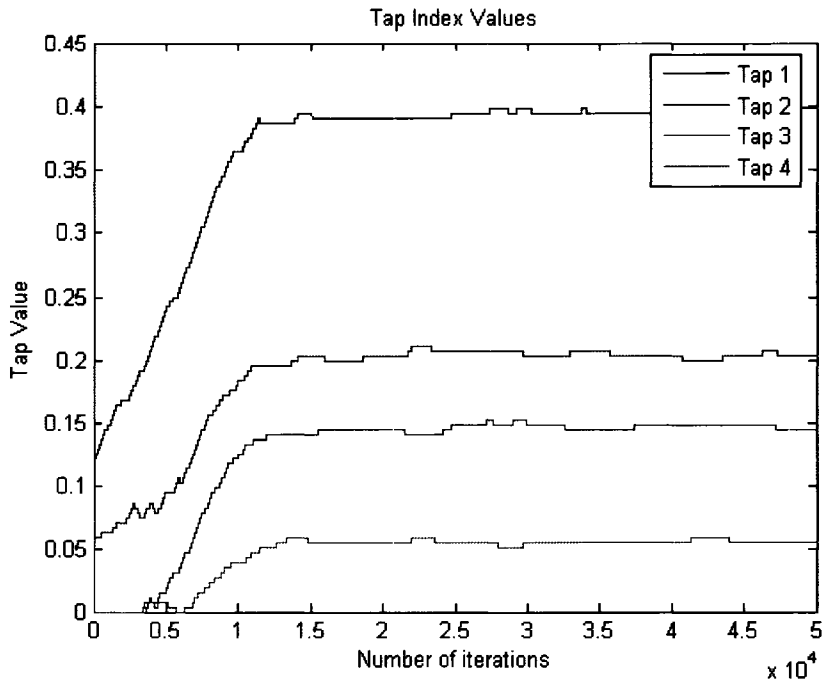

Figure 4.2 The Tap Convergence Without Training.

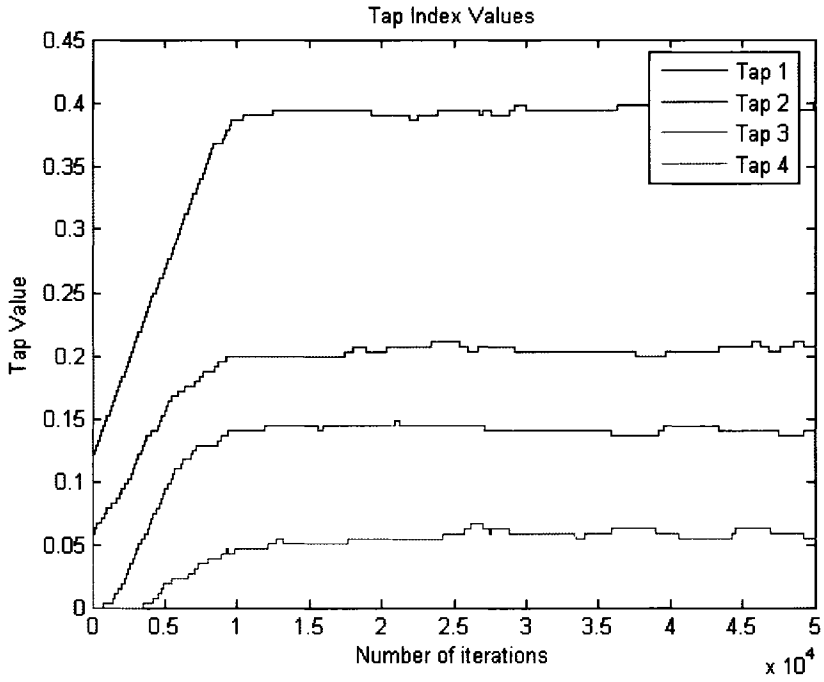

Figure 4.3 The Tap Convergence With Training.

With very little difference in the performance of the two systems, one must now look at the output of the DFE in order to determine whether or not the output is adequate for use by the data recovery circuitry. The figure 4.4 below shows the output eye-diagram from the DFE after the tap coefficients have converged to the above values. It is clear that the converged tap coefficients are correct since there are two clearly open eyes present in the output.

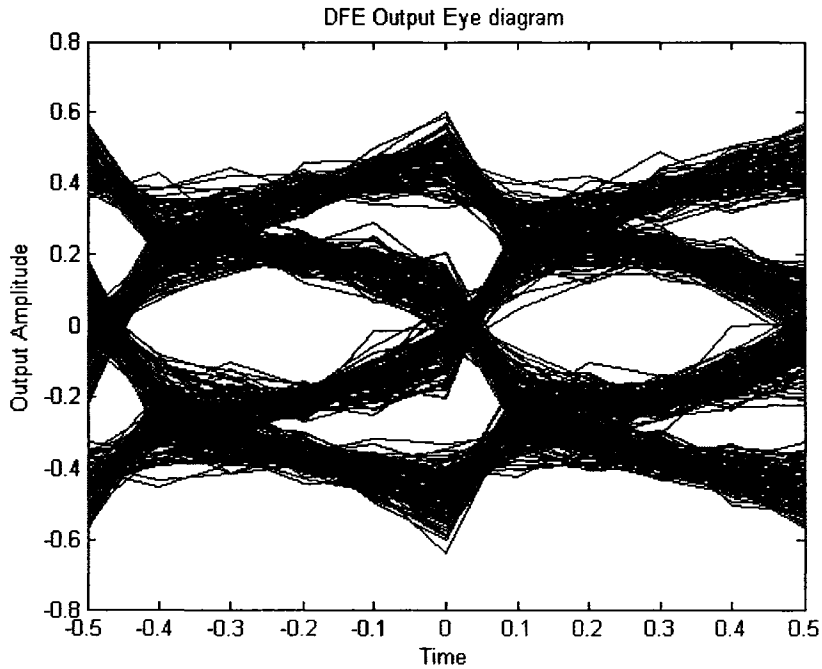

Figure 4.4 The Eye-Diagram at the Output of the DFE (in Volts).

Although the above eye-diagram shows significant improvement from the raw output of the channel, it does not show optimally equalized data symbols. This is because there is still a significant amount of pre-cursor ISI which remains unchanged since the DFE only counteracts post-cursor ISI. With the use of additional equalization methods, the output could be returned to the ideal levels. However, it is clear that the performance of the blind equalization method has been demonstrated in the above simulation.

4.4 Sensitivity to Model Parameters

Although the above simulation results show exceptional performance under the designed conditions, there are many solutions possible with the number of design variables as presented in the previous chapter. These variables allow for many unique solutions, each with a varying degree of performance and accuracy. The parameters evaluated in this section will include; adaptation block size, pre-counter threshold, and error threshold. Therefore, this section will show these design parameters used in the proposed adaptation method, their useful range, and their effect on the overall performance of the system. For each of the following simulations, both the trained and un-trained results will be shown in order to evaluate the reliability of the blind adaptation engine.

4.4.1 Block Size Sensitivity

The first analysis will be performed on the adaptation block size. As discussed in the previous chapter, the use of block adaptation allows the timing constraints on the adaptation engine to be significantly reduced. The block adaptation method also provides averaging in order to reduce the sensitivity of the tap coefficients to individual errors. This averaging will be one of the main factors in selecting the block size. The effective limit of the block size is determined by the slowest acceptable adaptation speed as defined by the system requirements. Conversely the smallest block size which would give the fastest possible adaptation speed, is limited by the stability of the algorithm as well as the maximum capabilities of the hardware implementation. Although these boundaries have not been defined in this work, the following simulations will show the possible block size values for which the algorithm is capable of effective operation.

The simulation results presented below in figures 4.5 and 4.6 show the convergence results for the adaptation engine using a range of block sizes. Rather than showing individual simulation results for each block size, the results have been summarized into two graphs. Each graph shows the four different tap coefficient values over a range of block sizes using the same colour scheme as the earlier figures with tap 1 in blue, tap 2, 3 and 4 in green, red and cyan respectively. For each block size, a simulation starting from the initial condition was performed and the coefficients were allowed to converge for 300,000 symbols. At the end of each simulation the tap coefficient values were averaged over the final 50,000 symbols and the results were plotted. In order to further gauge the performance at each block size, error bars were added to show the maximum and minimum values during the final convergence period. These error bars allow the final tap noise, which has been masked by the averaging, to be examined.

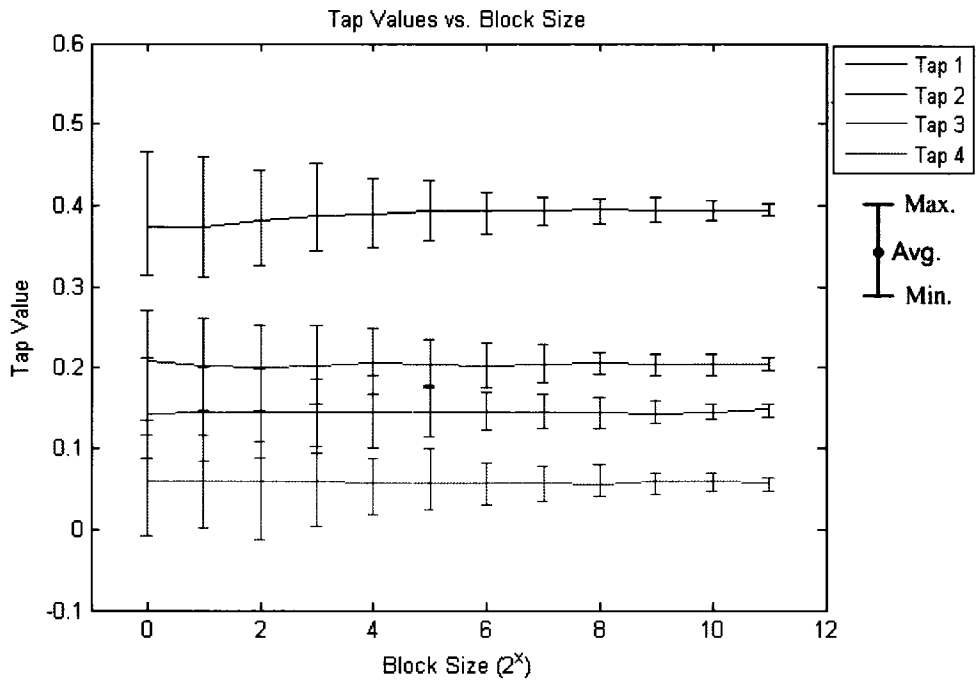

Figure 4.5 Sensitivity to Block Size (Blind)

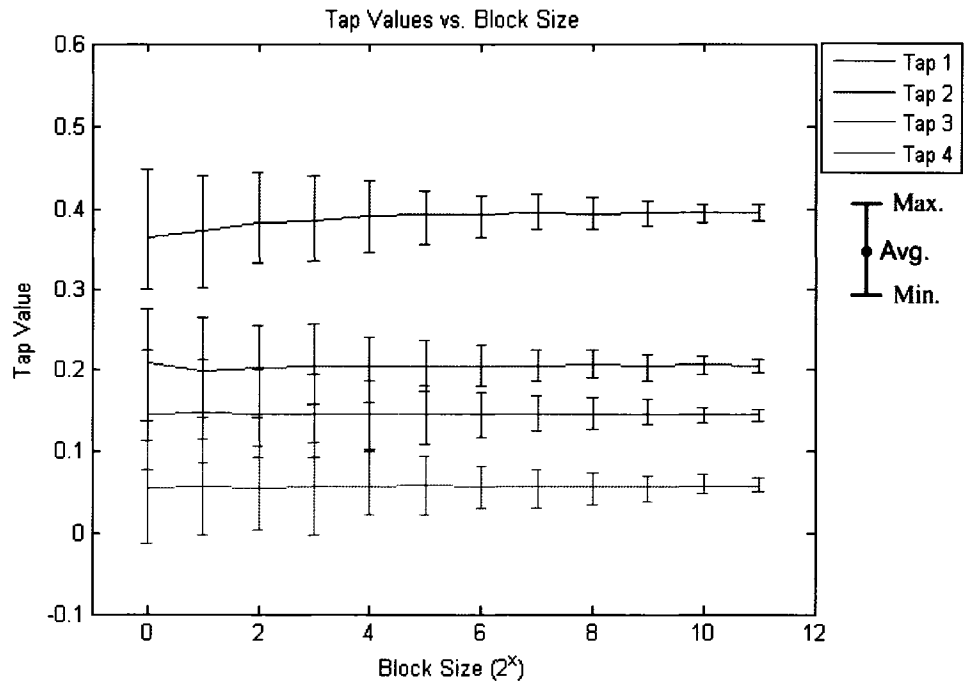

Figure 4.6 Sensitivity to Block Size (Trained)

It is clear from the above figures that both the blind and trained versions of the adaptation engine perform equally under various block sizes. It is also clear that small block sizes can affect the final convergence values as well as provide an unacceptable amount of tap noise on the coefficients. If the block size were taken beyond 2048 (2^{11}) the convergence time would increase beyond the length of the simulation. Since it is clear that the final convergence value remains unaffected by larger block sizes, further simulations would be fruitless. Therefore, a block size of 128 was selected in primarily to provide a short convergence period. With the block size set to 128 there is still a significant amount of tap noise which must be addressed. The figure 4.7 on the following page illustrates how the tap coefficients, although converged rapidly, exhibit an unacceptable amount of variation.

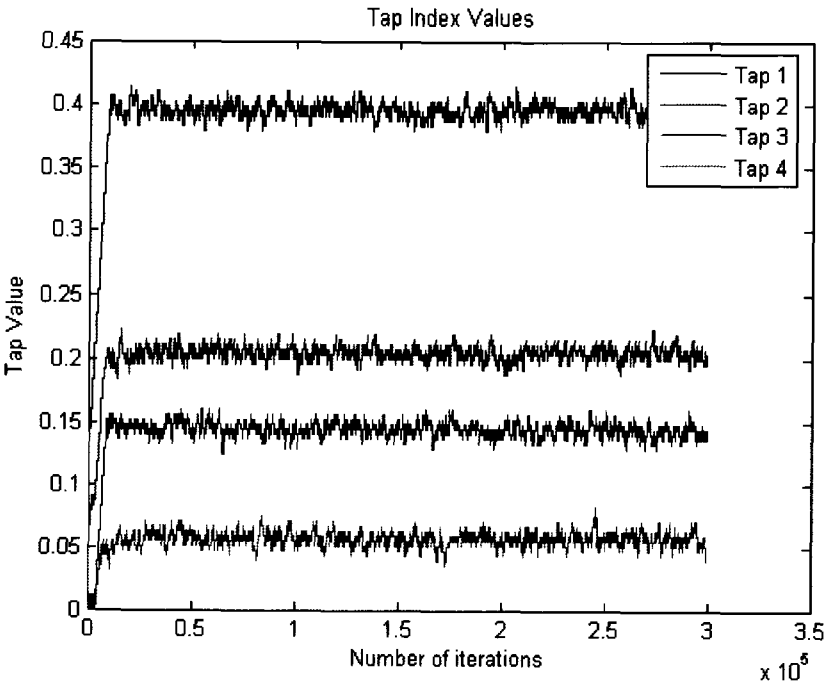

Figure 4.7 Tap Noise Without the Tap Pre-Counter Threshold.

4.4.2 Pre-Counter Threshold Sensitivity

As illustrated above, a separate technique is required in order to counteract the noise on the tap coefficients. Therefore, the tap pre-counter threshold will be introduced in order to prevent these numerous and unnecessary tap updates. It is extremely important however that the final convergence of the coefficients remain unaffected by this threshold since it is only intended to reduce noise. Using the block size of 128 symbols, as selected in the above section, an analysis was performed using various pre-counter threshold values. This analysis is similar to the one conducted in the previous section on the various block sizes. As in the above analysis, the results from both the trained and blind versions of the engine will be compared. Figures 4.8 and 4.9 on the following page show, how the tap coefficients of the two engines perform summarily under various pre-counter threshold values.

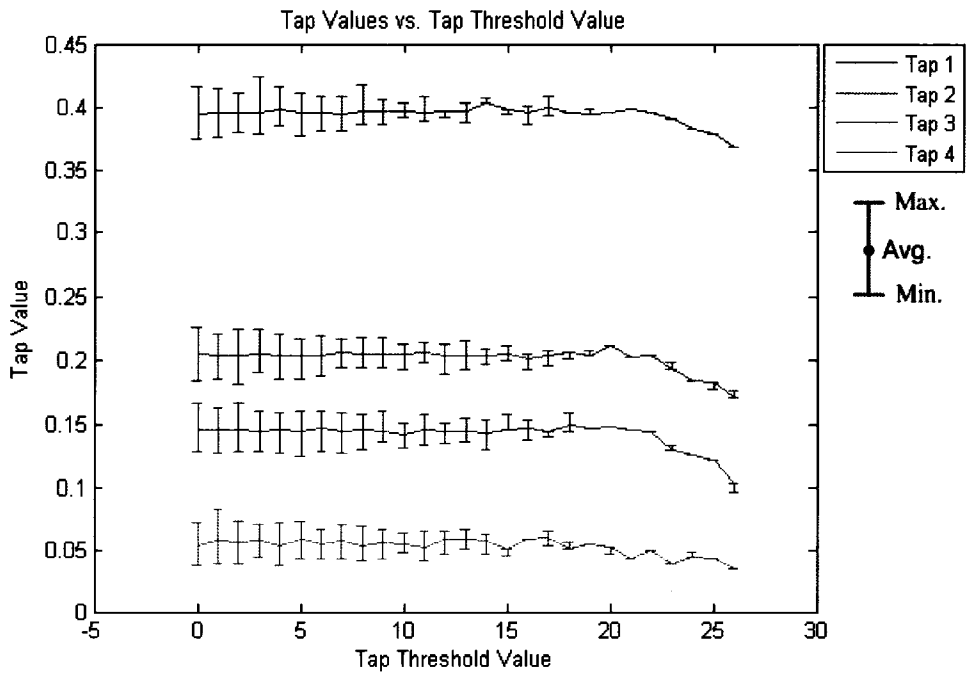

Figure 4.8 Tap Pre-Counter Threshold (Blind)

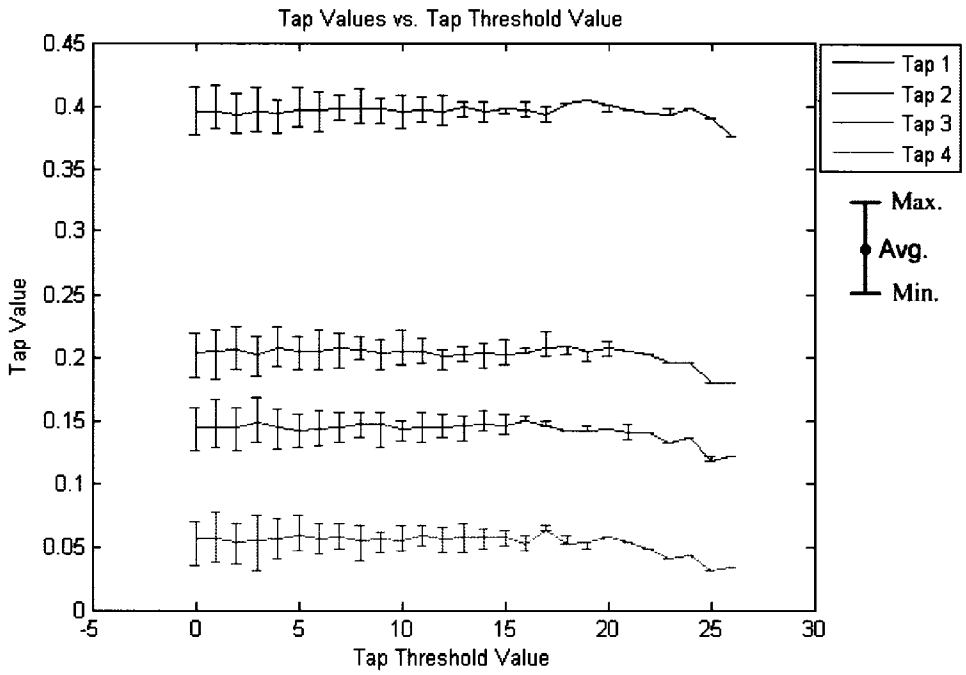

Figure 4.9 Tap Pre-Counter Threshold (Trained)

From the above figures it is clear that using a threshold in order to evaluate the tap pre-counter value is an effective technique to counter the residual tap noise. However, it should be noted that a value which is too large can affect the final convergence of the tap coefficients. Therefore, for the purposes of this design a pre-counter threshold of 8 was selected. Although a larger threshold value could be used, it is important that the adaptation engine is able to adjust the tap coefficients to changes in the environment. With a larger threshold value, the engine would require a more significant change in performance before adjusting the coefficients accordingly. It should also be noted here that the pre-counter threshold value is entirely dependant on the block size used. As the block size changes, the threshold would also have to change in order to remain effective against tap noise.

4.4.3 Error Threshold Sensitivity

The single most important variable in the design of the blind adaptive equalization engine is the error decision threshold. As mentioned in the earlier chapter, the error threshold is used to evaluate the quality of the incoming data symbols. If the symbol is beyond the threshold value it is assumed to be correct, otherwise if it is closer to zero it is assumed automatically to be in error. Therefore, it is clear that the error threshold value completely defines the operation of the adaptive engine. In order to properly determine the validity of the incoming symbols, it was obvious that the error threshold value would be dependant on the incoming symbol amplitude. The following figures 4.10 and 4.11 show the performance of both convergence engines with the error thresholds between 30% and 60% of the average incoming data symbol amplitude.

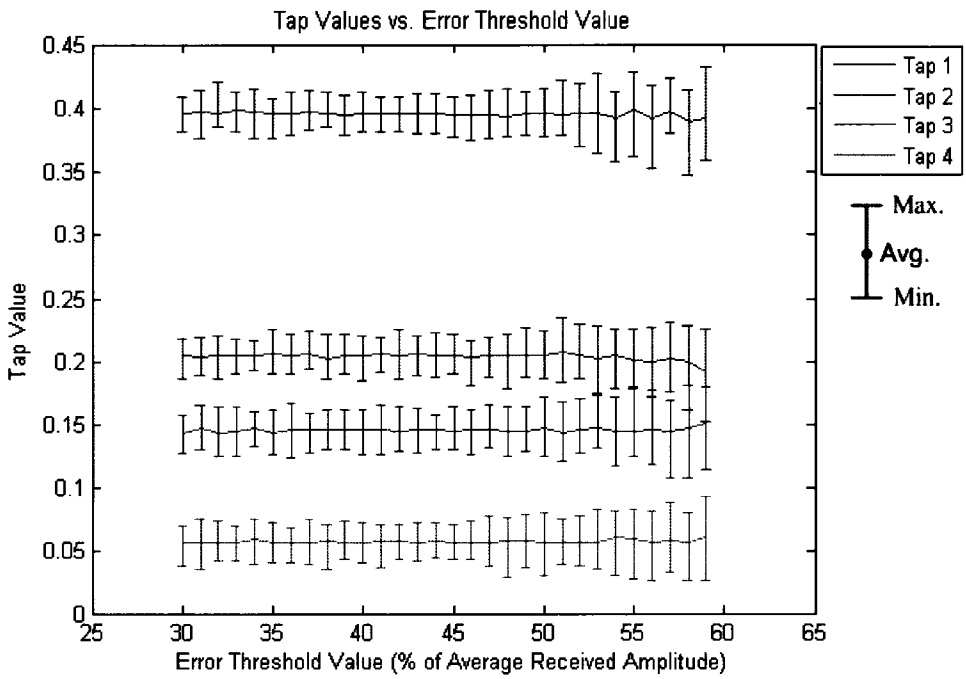

Figure 4.10 Error Threshold Sensitivity (Blind)

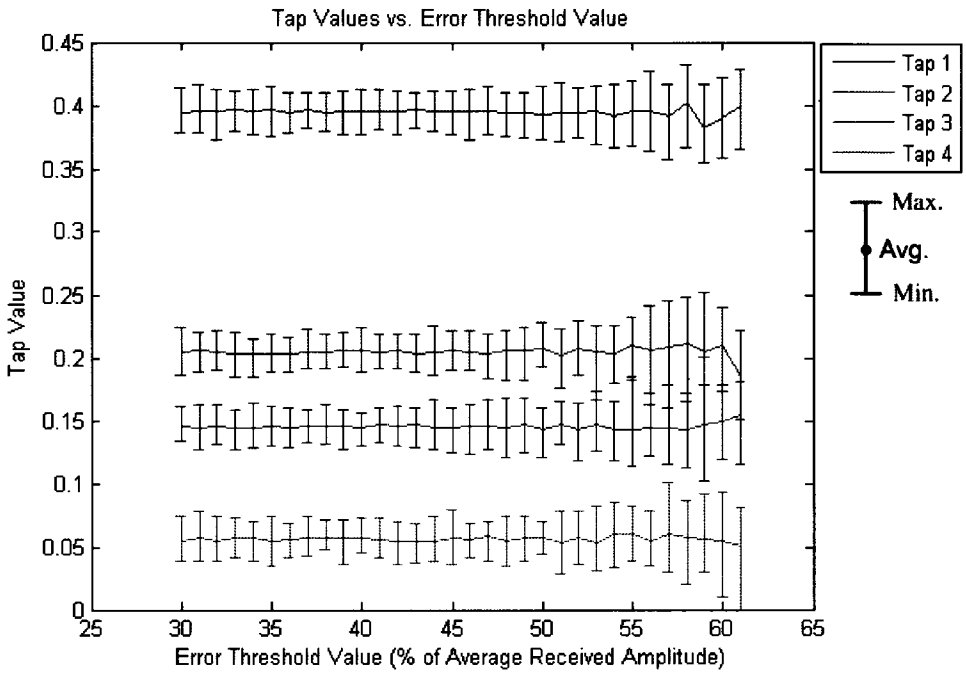

Figure 4.11 Error Threshold Sensitivity (Trained)

The purpose of the error threshold is to reduce the overall amount of error information being given to the adaptation algorithm while still permitting the more reliable information to direct the convergence. In the figures on the previous page, both the trained and untrained versions of the adaptation engine perform similarly under various tap threshold values. Both engines fail when the threshold falls below 30% of the input amplitude as there is not enough information provided to the algorithm to complete the adaptation. They fail once again when the threshold is brought above approximately 60% as there is too much insignificant error information and the algorithm begins to behave erratically. It should be noted here that the above simulations were performed with the pre-counter threshold set to zero as it is important to evaluate the raw performance under error threshold variation only. It is clear from the above simulations that the error threshold can be set anywhere within an approximately 30% window without significantly affecting the final convergence of the engine. Therefore, for reasons of reliability the error threshold will be set at approximately 45% of the incoming amplitude. This value will allow a buffer of approximately 15% on either side which will allow for any variation in the implementation of the decision circuitry or in the environmental conditions. One other factor which should be considered here is the performance of the engine in terms of adaptation speed. With an error threshold closer to the upper boundary of 60%, there would be a significantly greater amount of potentially valid error information being input into the algorithm. Although more error information would allow for rapid convergence of the tap coefficients, the robustness of the system would have to be sacrificed. Since correct operation of the system is of paramount concern, the error threshold will be based solely on the stability of the engine.

4.4.4 Window Size Sensitivity

As mentioned in the earlier sections, the error threshold which used to evaluate the quality of the incoming symbols, is based upon the incoming amplitude to the decision device. A specialized circuit will therefore be required to measure the overall amplitude and produce the reference signal at the desired level. This circuit could be based on a simple rectifier and RC type filter and will be discussed in further detail in the following chapter. However, this section of the proposed topology will help dictate the performance of the adaptation engine and its operation must be investigated further. From the above section, it is clear that the algorithm performance remains unaffected by threshold values which are above or below the target by up to 15%. Unfortunately this tolerance to amplitude error does not address the remaining issue of the RC time constant. The time constant of the filter will determine the time period over which the incoming amplitude will be measured. To model this effect, a windowing technique was used to measure the incoming amplitude. The size of this window dictates the number of past symbols that are considered in the measurement, the larger the window, the more past symbols are included in the measurement. Although this technique is rather simplified, it is effective in creating a direct comparison of the measurement window to the adaptation block size. This direct comparison is extremely important from perspective of algorithm stability. Since the measurement of the amplitude will be occurring simultaneously with the coefficient adaptation it is important that the two processes do not conflict with each other. To examine the interaction of these two processes, the block size was fixed to 128 symbols and the window size was varied from 0 (measuring only the current symbol) to 4096 (measuring more of the past symbols). The following figures 4.12 and 4.13 illustrate the performance of the tap adaptation process under various window size values for both the blind and trained versions of the engine.

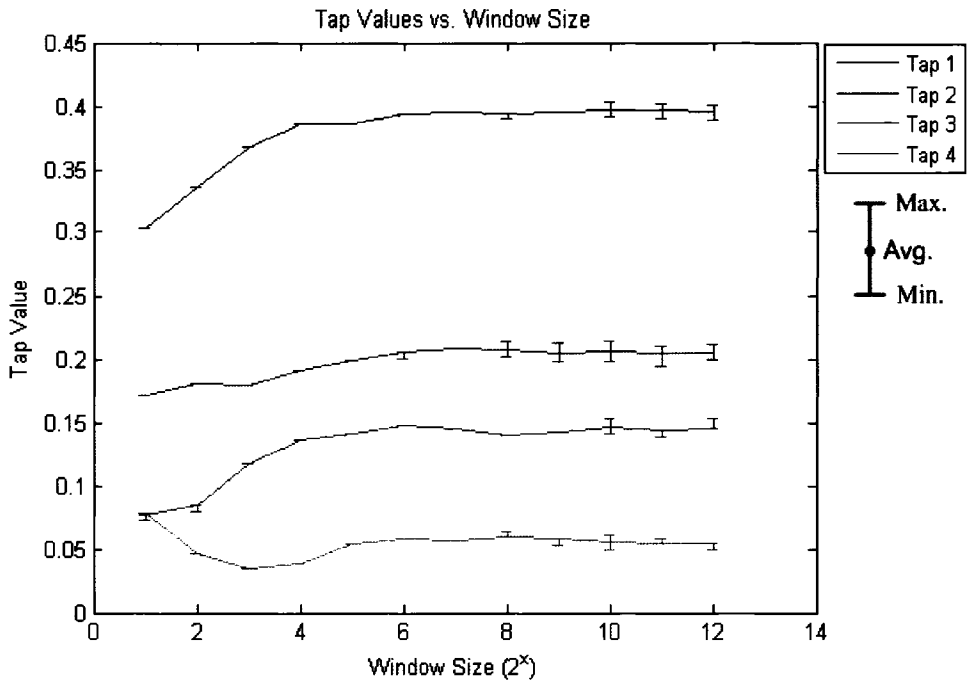

Figure 4.12 Measurement Window Size Sensitivity Analysis (Blind)

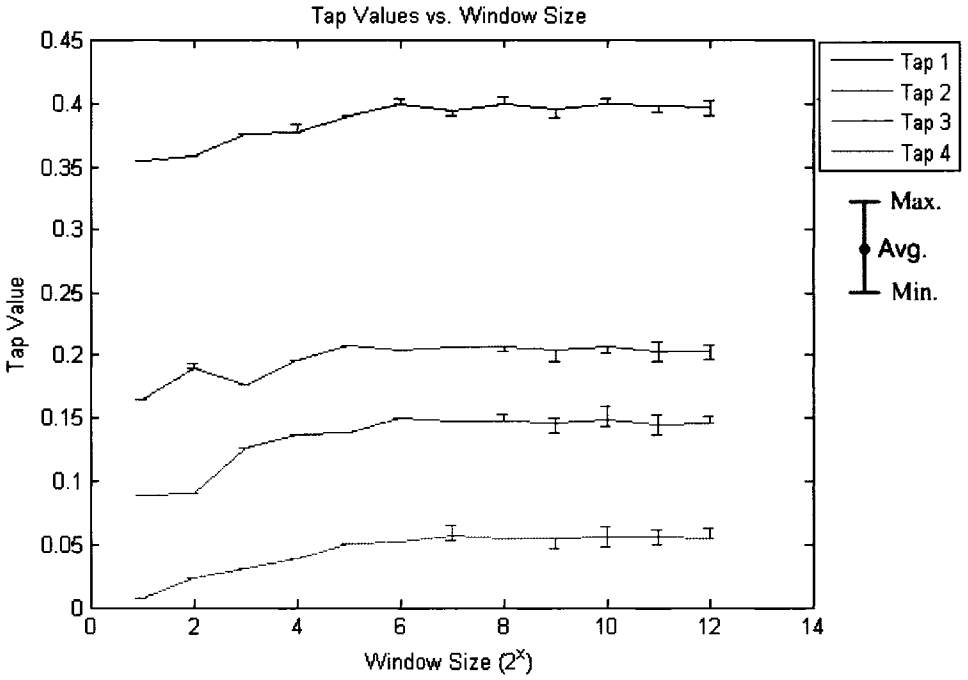

Figure 4.13 Measurement Window Size Sensitivity Analysis (Trained)

From the above simulation results, it is clear that window sizes closer to that of the adaptation block size interfere negatively with the performance of the adaptation process. At the same time, larger window sizes do not seem to negatively affect the final convergence values. Therefore it is important that the designer of the system ensures that the time constant of the filter is sufficiently long as to not affect the final convergence values.

4.5 Coefficient Noise Sensitivity

One of the most critical parts of the entire DFE and adaptation structure, is the summation circuit. This is the circuit which implements the core function of the filter by adding the symbols received from the channel to the symbols being fed back from the delay line of the filter. Any excess error at this node will interfere with the decisions being made causing errors which will affect the rest of the adaptation process. It is therefore important to verify wether or not the adaptation engine is able to tolerate a reasonable amount of this noise while still converging to an acceptable value. The resonable amout of noise in a structure similar to that quoted in this work, can approach 2% within each of the multiplication points of the DFE filter [42].

To test this capability, random noise was added to each of the feedback paths of the DFE filter. The variance of the amplitude of this noise was then swept from zero to approximately 70mV for both the trained and untrained versions of the algorithm. Since the full scale amplitude of the signals within this filter structure would be approximately 500mV, the error sweep is from 0% to 14%. It should also be noted here that the error due to quantization of the tap coefficients would be comparatively low. In the case of this proposed implementation, the tap quantization error would be approximately 0.8% which represents a precision of 7 bits.

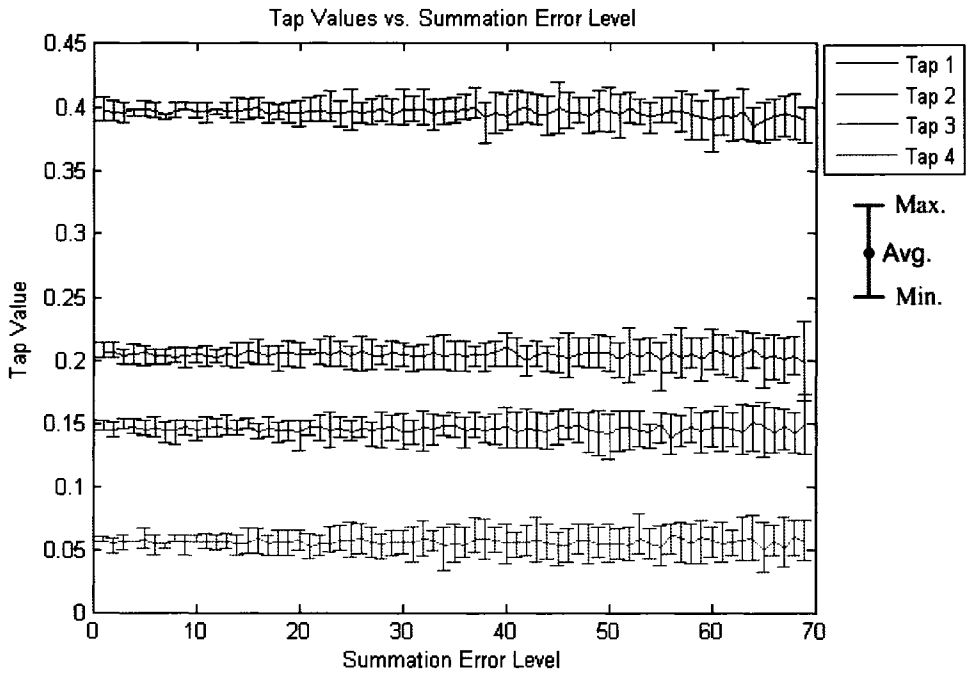

Figure 4.14 Tap Noise Sensitivity Analysis in mV. (Blind)

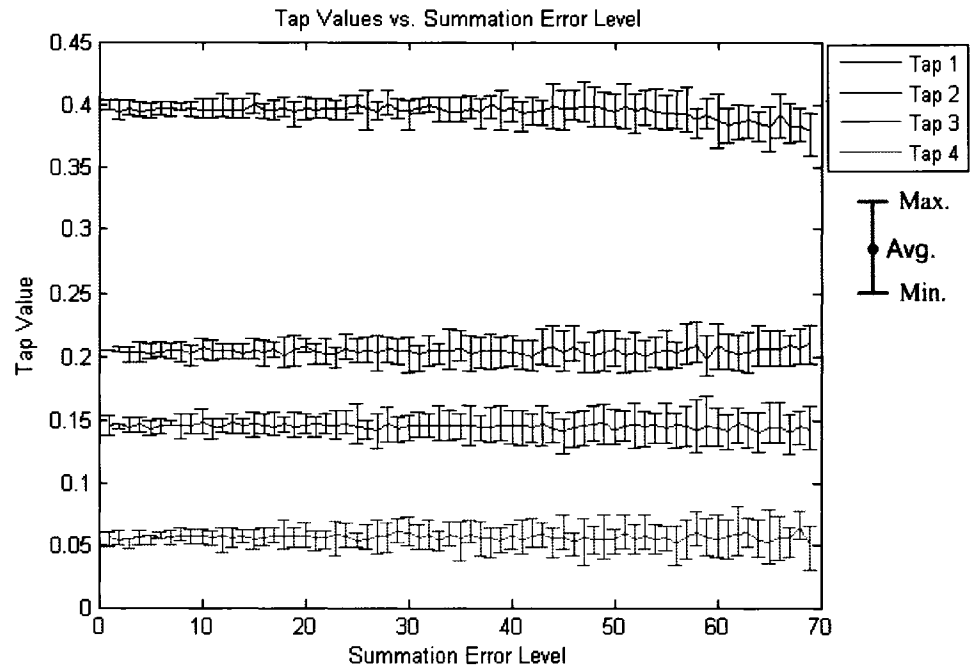

Figure 4.15 Tap Noise Sensitivity Analysis in mV. (Trained)

From the above performance it is clear that the algorithm can tolerate a significant amount of noise while still bringing the tap coefficients to an acceptable value. This property is especially valuable since there will be many sources of errors in the final circuit all of which could effect the final performance of the algorithm.

4.6 Summary

From the above simulation results, it is clear that this new method of Blind Adaptation is capable of converging the DFE coefficients to an acceptable level. Furthermore, it is capable of overcoming environmental challenges such as noise and various implementation choices which describe precision and the threshold levels. Unfortunately there is no physical circuit measurements that could be produced since it would be very difficult to design a receive-end equalizer without a full serial transmission system. Therefore this work will be limited to discussing the possible implementation in the following chapter and using the above simulation results to prove the feasibility of the design.

CHAPTER 5

Proposed Circuit Level Implementation

5.1 Introduction

Although proper operation of the blind adaptation algorithm has been fully demonstrated using Matlab simulations, one must demonstrate that it could be implemented into a real world circuit implementation. Unfortunately, due to the amount of complex circuitry required in a serial link transceiver it would be unfeasible to attempt a full circuit implementation within the time allotted. Furthermore, a partial implementation using purely digital circuitry would not yield any appreciable insight since the algorithm relies heavily on the operation of the forward analog circuitry as well as the filter structure. Therefore, to overcome these issues, the operational feasibility will be demonstrated using a combination of research and known topologies which have been proven in literature. This chapter will present two components critical to the proposed blind adaptive equalizer and will show the existence of equivalent operational topologies thereby verifying the feasibility and validity of the blind adaptation algorithm. The first component is the error detection circuitry, which identifies the symbols that will lead to a correction in tap value. The second is the high speed counter, which must keep track of the identified symbols and their polarity so that the algorithm may later decide on a possible tap update. In both instances, speed and accuracy are of utmost importance.

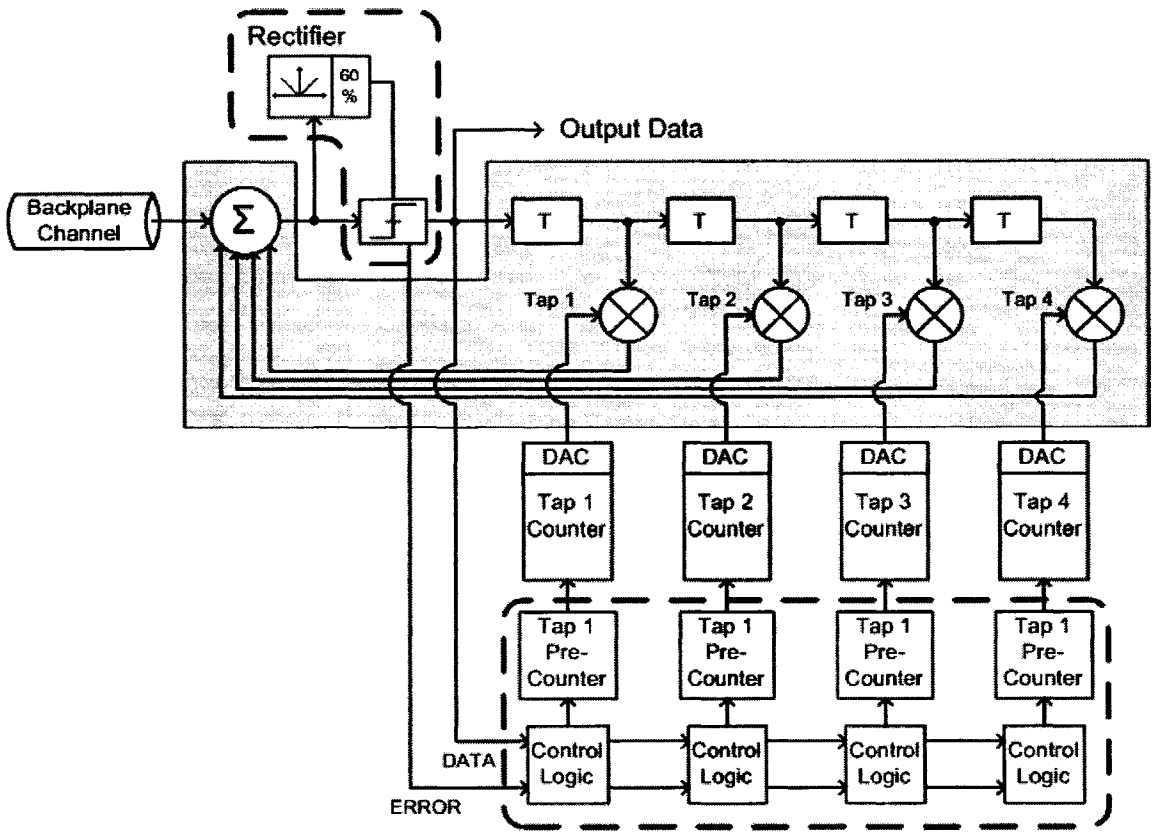

Figure 5.1 Topology of the DFE (Grey) and Adaptation Engine.

5.2 Overall Topology

The above Figure 5.1, shows an overview of the topology of the proposed DFE filter and adaptive equalizer algorithm. The area shaded in grey is the typical DFE filter structure and is not considered in this chapter since it is a standard topology that has been proven in literature. The areas of greatest concern are the sections highlighted in the dashed boxes. These are the components discussed earlier which require accurate high-speed operation. On the top left hand side of the diagram is the circuitry responsible for processing the incoming symbols, and making decisions on their value and validity. These components will be discussed in section 5.3. On the bottom of the diagram is the high-speed counter and control logic, which will be discussed in section 5.4. These two circuit blocks are key to the adaptation algorithm and would be the most difficult to construct.

5.3 Error Detection Circuitry

The most critical circuit to the successful operation of the blind adaptation algorithm is the error detection circuit. As stated earlier, this circuit is responsible for detecting whether or not the incoming bits are of sufficient amplitude to be relied upon as being correct. To function properly, this unit requires two parts; the first will create a relatively stable dc value based on the incoming amplitude. The second will compare this dc level with the amplitude of the individual bits, and issue a logical one if the bit is stronger than the reference value, and a logical zero if it is not. Both segments require somewhat precise analog circuitry in order to form the correct decisions at the required speed, since the algorithm relies on the error information directly in order to achieve convergence.

5.3.1 Error Decision

In order to distinguish whether or not an incoming symbol is greater or less than the reference value, a high-speed decision circuit is required. This type of circuit is quite common, as such a circuit would be required to differentiate between incoming zeros and ones.

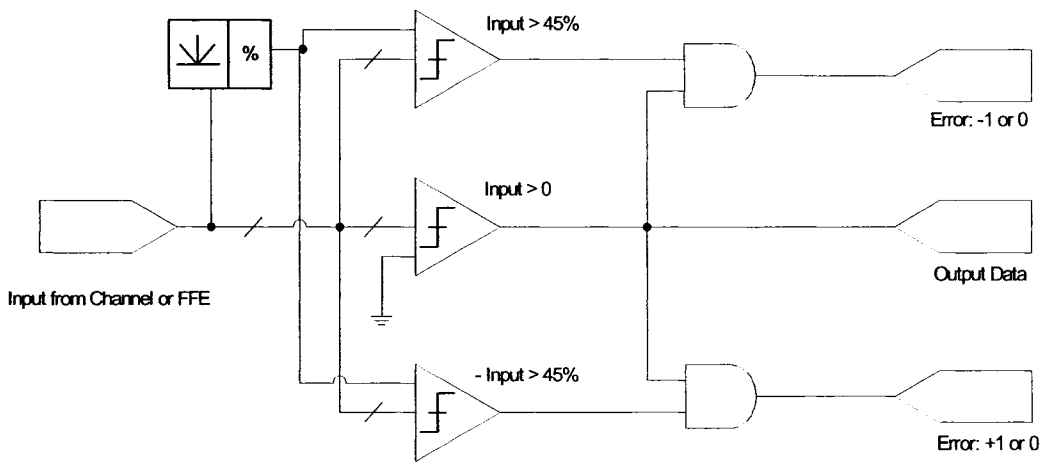

Figure 5.2 The Simplified Error Detector Topology.

Figure 5.2 shows the basic topology of the proposed error detection unit, three slicers are required to detect errors and the incoming data while a minimal amount of logic is required to interpret their outputs. Therefore, of great interest to this work are the several types of high speed slicers which have been designed for use in multi-level detection. Specifically those used in signalling schemes such as duobinary [2], and for four level PAM signalling [33]. In the case of the 4-PAM architecture seen in [33] a Common Mode Logic (CML) type sense amplifier is used for the individual detectors, while a Digital-to-Analog Converter (DAC) is used to synthesize the threshold voltages. In both cases, the topologies are employed within receiver structures capable of processing incoming serial binary data above 2.5Gb/s making them possible candidates for use in the algorithm proposed in this work.

5.3.2 Threshold Creation

In order to correctly identify the incoming bits into the receiver, a relatively stable DC value must first be created to form a comparison. The stable DC value in this case is approximately 45% of the incoming amplitude. In keeping with the adaptive nature of the receive end equalizer, the threshold value must be able to vary with time according to the incoming amplitude. To accomplish this stable value, a rectifier type circuit will be first used to extract the full amplitude of the incoming data. From this full amplitude, a much simpler circuit would be used to extract the threshold value. This could be as simple as a resistive load to divide the voltage. Such a circuit has been used recently by [38], in the context of a high speed adaptive equalizer environment. Specifically, it was used to in conjunction with a set of filters to extract the high and low frequency power. The precision of the rectified output was extremely important as the filter adaptation depended heavily

on an accurate comparison. Therefore such technique could be an excellent candidate for use in the blind adaptation method.

There may be several different techniques which could be used in order to extract the desired reference level from the incoming signals. However, to discuss these possible implementations would be fruitless without first defining the complete front-end topology of the receiver. Considering the accuracy required in order to ensure proper matching and any interface to other possible equalization schemes, further speculation would be inappropriate.

5.4 High-Speed Counter Design

The final operational requirement for the blind adaptive equalizer structure is a counter capable of keeping track of the incoming errors as identified by the error detection circuit. This counter would need to be able to operate at the same frequency as the incoming data. Fortunately, as described in chapter 3, this counter would only need to process up to 5 bits at a time since it would not be used to store the tap value itself, rather it would be used to accumulate the number of errors to help prevent erroneous tap updates. Due to the fact that the incoming errors contain polarity information, the counter would also need to perform subtraction in order to account for the direction of the possible tap update. Although such an adder would need to be custom designed there are several examples of large scale high performance architectures of this nature found in literature.

One circuit of interest was designed and reported by Intel Corporation for use in a high performance microprocessor. It can perform single-cycle 64-bit addition, subtraction and logic functions at a speed of 4GHz [34]. To achieve this level or performance, a special logarithmic circuit topology is used. Also, the circuit uses a specialized fabrication

process and consumes a significant amount of power. Fortunately, the addition/subtraction unit required here would not operate with such large numbers. Therefore, a smaller version of this topology would be an excellent candidate for implementation.

With any circuit topology that is selected, a high performance logic family would be required in order to perform at the desired clock frequency. An excellent reference for such high performance logic can be found with the design of counters/dividers, utilized in frequency synthesis Phase Locked Loop (PLL) circuits. These high-performance structures typically use MOS common mode logic (MCML) which allow the circuits to operate at a much higher frequency while consuming less power compared to conventional CMOS logic. This is accomplished through the use of a constant current source and differential pairs of MOS transistors which switch the current between two load resistors [35]. Therefore a relatively constant amount of power is used regardless of frequency. A recent example of MCML logic use in the construction of a high-speed divider can be found in [36], which reports a 256/257 dual-modulus prescalar designed in 120 nm CMOS technology and can operate at 15 GHz. The key factor in the circuit which allows for such high frequency operation, is the specially designed MCML merged AND-gate flip-flops. These merged gates, seen in Figure 5.4, allow a reduction in area and power consumption with a reduction in delay, thus allowing high speed operation.

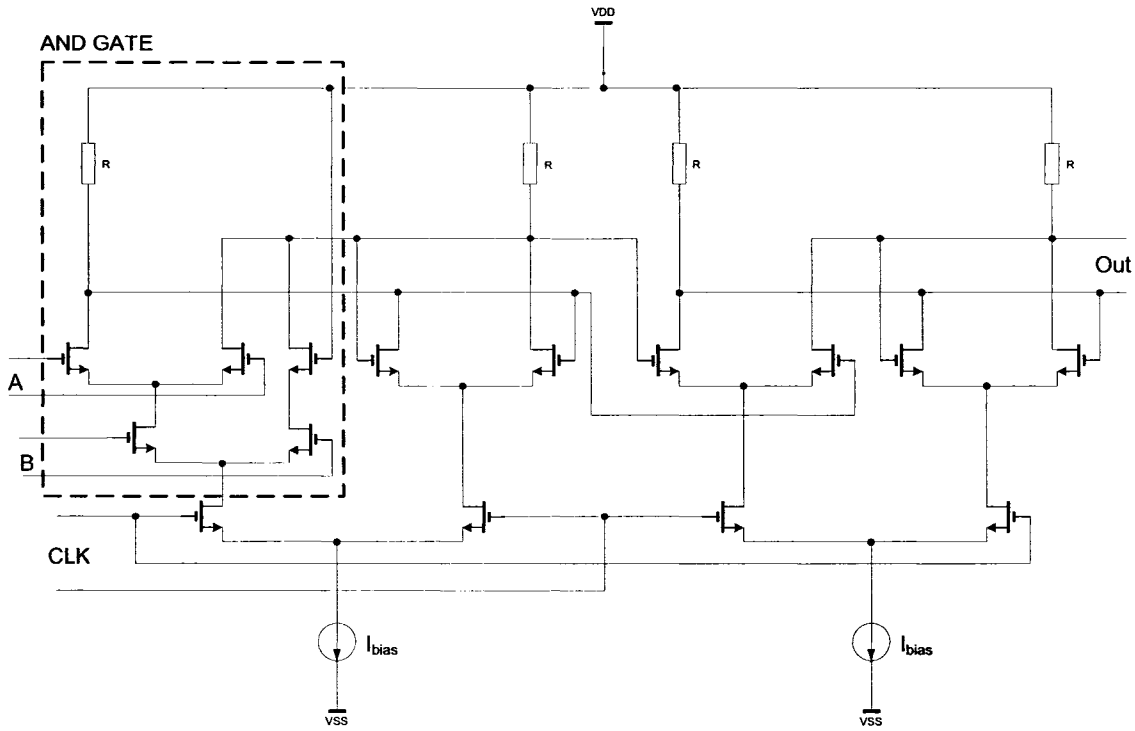

Figure 5.3 The Merged AND gate Flip-Flop from [36]

The speed of this structure could be further advanced with the use of a more advanced technology or circuit technique. A second topology reported by the same research group uses the more advanced 90nm CMOS technology node as well as inductive peaking in order to increase the bandwidth and voltage head-room limitations [37]. The circuit can divide the incoming signal by 4 or 5 and can operate at a frequency of 24GHz. Although the requirements for the blind adaptation method are far below this level, the techniques and designs presented here show the feasibility of larger, high speed logic functions such as the addition/subtraction unit.

5.5 Area and Power estimates

The required area and power for the proposed Blind Adaptation algorithm will be similar to that required for a conventional trained system. In the previous work, which this work is based upon, completed by Lei Lin, a pre-emphasis FIR adaptation engine is designed and measured for a 5 Gbps serial link across the same 34-inch backplane. This design contained the following parameters.

TABLE 1. Area and Power used in design [5]

Item	Measure
Clock Frequency	500 MHz
Total Cell Area	113608um^2
Total Dynamic Power	49.3mW

5.6 Summary

With the topologies described above, it is clear that the method of blind adaptive equalization as presented in this work is feasible within standard CMOS circuitry. With the use of current mode circuitry and an advanced CMOS process such as 90nm the circuits described could be made to operate at the required signalling speed. Furthermore, the circuits presented in this chapter would be accurate enough to allow for the proper operation of the blind adaptive algorithm. Although further work would need to be done in order to construct these circuits, it is not necessary since they have already been proven in literature and through other research done at Carleton. Unfortunately, the design and construction of these circuits would not be possible within the time allotted for a Masters thesis.

CHAPTER 6

Conclusion and Future Work

6.1 Summary

Currently, most DFE structures rely on a specialized sequence of data which must be used to properly adapt the coefficient values. In high-capacity serial backplanes, this training sequence can waste valuable time as well as cause interoperability issues. It has therefore been the goal of this research work to prove that a Decision Feedback Equalizer structure is able to operate effectively without the use of a training sequence. Through the use of MATLAB simulations and channel measurement data, a standard algorithm has been adapted to use an alternative method for ensuring the reliability of its operation. This reliability has been shown in numerous simulations which subject the proposed method to alternate design choices as well as circuit noise.

This work is firmly based on the measured response of a typical serial backplane environment. Furthermore, this work makes use of a standard algorithm commonly used in many modern DFE implementations as a solid foundation for the research work. This foundation is combined with the analysis of the behavior of the system against various design parameters and noise which would be consistent with that found in a CMOS implementation.

6.2 Contributions

The main contribution of this thesis has been the modification of a known adaptation algorithm in order to remove the training sequence used during the initial startup of a DFE. These modifications include the use of a secondary threshold as a measure of the quality of signals coming from a serial backplane channel and how this newly qualified data interacts with the adaptation algorithm. Previously reported, DFE implementations have relied on a training sequence being transmitted across the backplane during start-up in order to set the tap coefficient values of the DFE. It is believed that this solution could be used as a self-adaptive DFE system for any ISI limited channel.

6.3 Future Work

Although this work has proven that it is possible for a DFE to operate without the use of a training sequence, the most effective and definitive proof would be implementation. Therefore, the remaining work involved in this research endeavour would be to incorporate the proposed architecture within a compatible DFE CMOS implementation for use in a high-speed serial environment similar to the model used here. Furthermore, the operation of this proposed adaptation scheme could be considered for use with other emerging techniques such as edge or transition based equalization. Furthermore this technique may be combined with forward error correcting schemes in order to explore their interaction and possible collaboration during the adaptation phase.

References

1. F. Grisin and Z. Pantic-Tanner, "Design advances in PCB/Backplane interconnects for the propagation of high-speed Gb/s digital signals," *IEEE TELSIKS 2003*, vol. 1, pp. 184-191, October 2003.
2. J.H. Sinsky, M. Duelk and A. Adamiecki, "High-Speed Electrical Backplane Transmission Using Duobinary Signaling" *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, No. 1, January 2005.
3. C. E. Berndt and T. Kwasniewski, "A Review of Common Receive-End Adaptive Equalization Schemes and Algorithms for a High-Speed Serial Backplane". *Proceedings of the 5th IEEE Workshop on System-on-Chip for Real-Time Applications*, July 2005, pp. 149-153.
4. T. Granberg, *Handbook of Digital Techniques for High-Speed Design.* Upper Saddle River, NJ, U.S.A: Prentice-Hall, 2004. pp. 31-45
5. Lei Lin, "A VLSI Implementation of an Adaptation Algorithm for a Pre-Emphasis in a Backplane Transceiver". *M.A.Sc. Thesis,* Dept. of Electronics, Carleton University, 2003.
6. C.Pelard, *et al.*, "Realization of Multigigabit Channel Equalization and Crosstalk Cancellation Integrated Circuits," *IEEE Journal of Solid-State Circuits.* vol. 39, No. 10, pp. 1659-1670, October 2004.
7. S. Qureshi, "Adaptive Equalization," *IEEE Communications Magazine*, vol. 20, Issue: 2, pp. 9-16, March 1982.

8. A.J. Kim *et al.*, "Equalization and the Evolution of Gigabit Communications," *IEEE GaAs Digest 2003,* pp. 193-196, Nov. 2003.

9. K. Lazaris-Brunner, and J. D'Ambrosia, "10-Gigabit Backplanes: Active-Copper-backplane interconnects go faster and farther," *EDN,* February 5, 2002.

10. J. Yang, *et al.*, "A Quad-Channel 3.125Gb/s/ch Serial-Link Transceiver with Mixed-Mode Adaptive Equalizer in 0.18um CMOS," *IEEE International Solid-State Circuits Conference,* Feb. 2004, pp. 176-520

11. Y. Hur, *et al.*, "Equalization and Near-End Crosstalk (NEXT) Noise Cancellation for 20-Gb/s 4-PAM Backplane Serial I/O Interconnections" *IEEE Transactions on Microwave Theory and Techniques,* Vol. 53, No. 1, pp. 246-255, January 2005.

12. V. Balan, *et al.*, "A 4.8-6.4 Gbps Serial Link for Backplane Applications using Decision Feedback Equalization," *IEEE Custom Integrated Circuits Conference Proceedings,* October 2004, pp. 31-34.

13. S. Hoyos, J.A. Garcia, and G.R. Arce, "Mixed-Signal Equalization Architectures for Printed Circuit Board Channels," *IEEE Transactions on Circuits and Systems-I: Regular Papers,* Vol. 51, No. 2, pp. 264-274, February 2004.

14. G. Balamurugan, *et. al.*, "Receiver Adaptation and System Characterization of an 8Gbps Source-Synchronous I/O Link using On-die Circuits in 0.13um CMOS," *IEEE 2004 Symposium On VLSI Circuits Digest of Technical Papers*, pp. 356-359

15. R. Payne, *et. al.*, "A 6.25Gb/s Binary Adaptive DFE with First Post-Cursor Tap Cancellation for Serial Backplane Communications," *2005 IEEE International Solid-State Circuits Conference,* p. 68

16. S. Haykin, *Adaptive Filter Theory.* Englewood Cliffs, NJ: Prentice-Hall, 2002

17. P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation.* Boston: Kluwer Academic Publishers, 1997

18. S. Haykin, B. Widrow, *Least-Mean-Square Adaptive Filters.* Wiley Interscience, 2003.

19. M. Q. Le, P. J. Hurst, J. P. Keane, "An Adaptive Analog Noise-Predictive Decision-Feedback Equalizer". *IEEE Journal of Solid-State Circuits,* Vol. 37, No. 2, pp. 105-113, February 2002.

20. M. Q. Le, P. J. Hurst, K. C. Dyer, "An Analog DFE for Disk Drives Using a Mixed-Signal Integrator", *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 5, pp. 592-598, May 1999.

21. N. Krishnapura, "A 5Gb/s NRZ Transceiver with Adaptive Equalization for Backplane Transmission," *IEEE International Solid-State Circuits Conference,* 2005, p. 60.

22. B. Daneshrad, "HIPERLAN Equalizer ASIC Complexity and its relationship with the training header," *Journal of Wireless Personal Communications special issue on HIPERLAN,* Vol. 3, No. 4, pp. 421-426, 1996.

23. D. Wulich, A. Geva, "On a Periodic Training Sequence in DFE to Reduce the Steady-State Error Probability," *IEEE Transactions on Communications,* Vol. 47, No. 9, pp. 1288-1292, September 1999.

24. R. Payne *et al.*, "A 6.25-Gb/s Binary Transceiver in 0.13-um CMOS for Serial Data Transmission Across High Loss Legacy Backplane Channels," *IEEE Journal of Solid-State Circuits,* Vol. 40, No. 12, pp. 2646-2657, December 2005.

25. T. Beukema *et al.*, "A 6.4-Gb/s CMOS SerDes Core With Feed-Forward and Decision-Feedback Equalization" *IEEE Journal of Solid-State Circuits,* Vol. 40, No. 12, pp. 2633-2645, December 2005.

26. Maxim Integrated Products, "Designing a simple, small, wide-band and low power equalizer for FR4 copper links," *DesignCon, 2003.*

27. M. Li, S. Wang, T. Kwasniewski, "DFE Architectures for high-speed backplane applications," *Electronics Letters,* Vol. 41, No. 20, 29th September 2005.

28. M. Mizuno *et al.*, "A GHz MOS Adaptive Pipeline Technique Using MOS Current-Mode Logic," *IEEE Journal of Solid-State Circuits,* Vol. 31, No. 6, pp. 784-791, June 1996.

29. J.M. Musicer, J. Rabaey, "MOS Current Mode Logic for Low Power, Low Noise CORDIC Computation in Mixed-Signal Environments" *Proceedings of the 2000 International Symposium on Low Power Electronics and Design, 2000.* ISPLED '00. July 2000, pp. 102-107.

30. J. Jaussi *et al.*, "A 20Gb/s Embedded Clock Transceiver in 90nm CMOS" *2006 IEEE International Solid-State Circuits Conference,* pp. 340-341.

31. C. Kromer *et al.*, "A 25Gb/s CDR in 90nm CMOS for High-Density Interconnects" *2006 IEEE International Solid-State Circuits Conference,* pp. 326-327.

32. A. Rylyakov *et al.*, "A 30Gb/s 1:4 Demultiplexer in 0.12um CMOS" *2003 IEEE International Solid-State Circuits Conference,* Paper 10.2.

33. J. L. Zerbe *et al.*, "Equalization and Clock Recovery for a 2.5-10-Gb/s 2-PAM/4-PAM Backplane Transceiver Cell". *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 12, pp. 2121-2130, December 2003.

34. S.K Matthew *et al.*, "A 4-GHz 300mW 64-bit Integer Execution ALU With Dual Supply Voltages in 90-nm CMOS" *IEEE Journal of Solid-State Circuits*, Vol. 40, No. 1, pp. 44-51, January 2005.

35. M. Yamashina and H. Yamada, "An MOS Current Mode Logic (MCML) Circuit for Low-Power Sub-GHz Processors" *Trans. IEICE Electron.*, Vol. E75-C, No. 10, pp. 1181-1187, October 1992.

36. H.D. Wohlmuth and D. Kehrer, "A 15GHz 256/257 Dual-Modulus Prescalar in 120nm CMOS", *European Solid-State Circuits Conference,* pp.77-80, Estoril, Portugal, September 2003.

37. H.D. Wohlmuth and D. Kehrer, "A 24GHz Dual-Modulus Prescalar in 90nm CMOS" *IEEE International Symposium on Circuits and Systems 2005*, Vol. 4, May 23-26 2005, pp. 3227-3230.

38. J-S. Choi, M-S. Hwang and D-K. Jeong, "A 0.18-um CMOS 3.5-Gb/s Continuous-Time Adaptive Cable Equalizer Using Enhanced Low-Frequency Gain Control Method" *IEEE Journal of Solid State Circuits,* Vol. 39, No. 3, pp. 419-425, March 2004.

39. Zhenhua Wang, "Full-Wave Precision Rectification that is Performed in Current Domain and Very Suitable for CMOS Implementation", *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Applications,* Vol. 39, No. 6, pp. 456-462, June 1992.

40. Optical Inter-networking Fourm, "Signal Over Backplane" OIF 2004.004.01.

41. M. Meghelli *et al.*, "A 10Gb/s 5-Tap-DFE/4-Tap-FFE Transceiver in 90nm CMOS" *2006 IEEE International Solid-State Circuits Conference,* pp. 80-81.

42. Miao Li, "Design Methodology of Pre-Emphasis Filtering for Digital Data Communications". *M.A.Sc. Thesis,* Dept. of Electronics, Carleton University, 2003.