A LOW-POWER HIGH-SPEED DIGITAL ADAPTIVE EQUALIZER FOR MAGNETIC DISK DRIVE CHANNELS UTILIZING CLASS IV PARTIAL RESPONSE SIGNALLING

by

Jacques C. Rudell

Memorandum No. UCB/ERL M94/14

14 March 1994

A LOW-POWER HIGH-SPEED DIGITAL ADAPTIVE EQUALIZER FOR MAGNETIC DISK DRIVE CHANNELS UTILIZING CLASS IV PARTIAL RESPONSE SIGNALLING

by

Jacques C. Rudell

Memorandum No. UCB/ERL M94/14

14 March 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A LOW-POWER HIGH-SPEED DIGITAL ADAPTIVE EQUALIZER FOR MAGNETIC DISK DRIVE CHANNELS UTILIZING CLASS IV PARTIAL RESPONSE SIGNALLING

by

Jacques C. Rudell

Memorandum No. UCB/ERL M94/14

14 March 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Low-Power High-Speed Digital Adaptive Equalizer for Magnetic Disk Drive Channels Utilizing Class IV Partial Response Signalling

Abstract

A novel approach utilizing pipelining and parallelism is used to implement a digital adaptive equalizer for a magnetic disk drive read channel employing Class IV partial response is described. The sign LMS (Least Mean Squared) Algorithm was used to update the filter tap coefficients. Data rates of a 100Mbits/sec were achieved while consuming less than 340mWatts of power. The architecture was fabricated on a prototype integrated circuit in a HP 1.2 μm CMOS MOSIS process. Details of the design and test results are covered in this thesis.

Acknowledgments

Because there are so many it is difficult for me to express my sincere thanks to all of those who have helped make this project a reality. Obviously, the success of this thesis would not have been possible without the guidance of my advisor Professor Paul R. Gray. Beyond support on many technical issues he also inspired me through many motivating conversations and I thank him.

To date, I have had several jobs both in industry and at academic institutions. In each of these positions I have had the opportunity to work with people who not only inspire me to do my best but also became a close friend, Greg Uehara is one such person. Words cannot possibly express my appreciation for all that he has done for me. Greg, now a Professor at the University of Hawaii, spent many long hours with me aiding in simulation and discussing design ideas related to this project. Beyond the DSP, Greg also took the time to answer many questions related to my course work and was instrumental in helping me prepare for my prelims. Caesar Wong, a fellow graduate student, now an engineer at Rockwell International, was responsible for getting this program off the ground and keeping it alive while I was preoccupied with another IC project unrelated to the DSP described in this thesis. More than any of us Caesar spent many long hours into the evening designing and verifying the four finite impulse response filters which comprise the heart of this chip. In addition to his contributions, Caesar's humor and quick wit made the long hours a little less tedious and little more light hearted.

Other students in Professor Gray's group made key contributions. In particular, Cormac "Dr. Pipe" Conroy, now at IBM Santa Clara, coached me through the design and layout of our test-board. I also shared many informative conversations with Keith Onodara, Todd Weigandt, Dave Cline and Thomas Cho. Other contributions also came from some of the newer students, Andy Abo, and Srenik Metha.

It almost goes without saying that none of this could have happened without the encouragement of my family, in particular my parents.

This work was support by the National Science Foundation, National Semiconductor, and the California Micro Program.

CHAPTER 1

Introduction

1.0 Introduction

At the youthful and naive age of fifteen, I remember working with a friend on a computer program that would simulate the hottest new game around called space invaders. At the time, when we were done working for the day we would store our program on a tape using a cassette recorder. While storing the game which consisted of 300 lines of code, we always had plenty of time to venture off in the kitchen and grab a glass of soda before the tape was finished. Thirteen years later at the more mature and informed age of twenty-eight I find myself writing a masters thesis using a MacIIsi which has the capability of storing entire sections of this paper with the blink of an eye. This drastic increase in the ability to store information on a decreasing amount of space can be attributed to the advances made in the area of magnetic storage. With the advent of the information age there has been an increasing need to find new methods to increase the amount of information in a given space. Accordingly, technologies in the field of information storage have become increasingly important. In 1992 the storage industry was a $50 billion a year business and is expected to grow to $100 billion by the year 1995. With this motivation many companies and universities are concentrating on methods to increase the storage density of magnetic media.

The emphasis of this thesis is to present a design done at Berkeley for the read electronics in a magnetic disk drive channel employing a signaling technique referred to as partial response. Specifically, high speed and low power were the two issues addressed by this research. The goal was to design a digital signal processing chip which would equalize samples coming from the magnetic disk drive channel at 100MHz while utilizing less than 300mWatts of power. It was further desired to make this filter adaptive to varying channel conditions.

The chip described in this thesis evolved from a larger project which was an attempt to create an entire high speed magnetic disk drive read channel. Basically, a two chip solution was required for our implementation of the read channel. The block diagram shown in figure 1.1 highlights the key features of the Berkeley system. The high speed Analog front end chip utilizes parallel processing to increase the overall throughput of the channel. Specifically, three parallel channels each running at 33MHz create an effective throughput of 100MHz. The analog front-end was a portion of a Ph.D project by Greg Uehara [16] and will be refer to from this point on as Uehara's front-end chip. Essentially, pulses from the read coil are passed into the Uehara chip which first performs a continuous time analog lowpass filtering operation. The analog pulses are then sampled and passed into a lowpass decimation finite impulse response filter realized using a switch capacitor filter circuit. The output of the FIR filter is then passed into a high speed analog to digital converter and then driven off-chip. Now the three parallel channels each running at 33MHz are then converted to two parallel samples running at 50MHz using commercial parts on a printed circuit board. The 50MHz samples are inputted to the DSP described in this thesis. Samples are then passed through a digital adaptive equalizer and then into a Viterbi sequence detector which outputs recovered data.

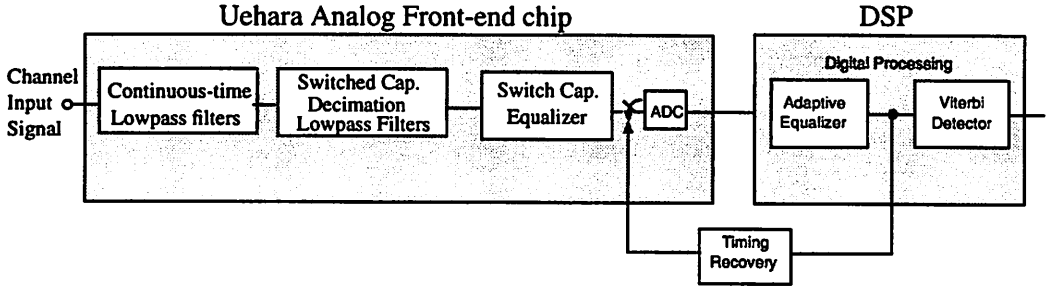


Figure 1.1 Block Diagram of the Berkeley read channel

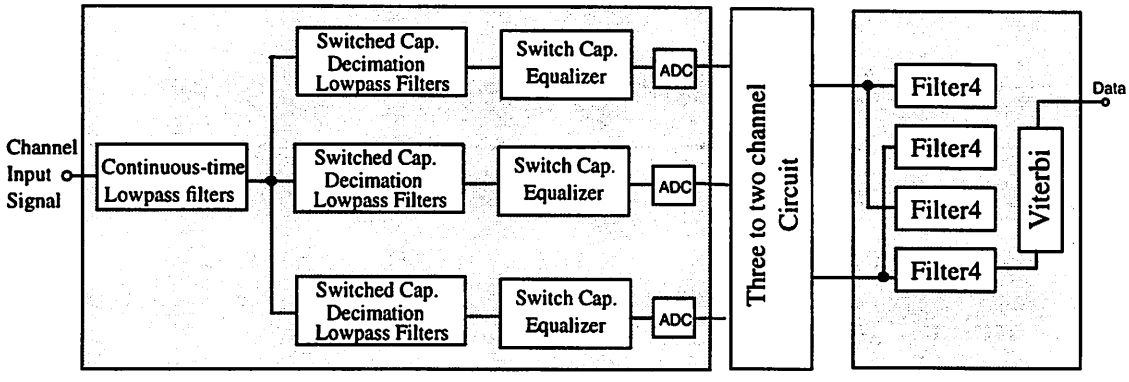


Figure 1.2 Block Diagram illustrating the parallel nature of the read channel.

The design and layout of the analog front-end chip was done as a Ph.D. project by Greg Uehara for further information consult his thesis [16]. The remainder of this thesis will concentrate on the specific details of the digital adaptive equalizer. In particular, chapter2 will cover the system design issues, simulation and description. Chapter 3 will focus on the description of the DSP architecture while chapter 4 addresses the specific issues surrounding the design of all the circuit blocks found on the DSP. A description of the test setup and measurement techniques are outlined in chapter 5 along with a description of a custom board built to interface the analog front-end with the DSP. This thesis will then conclude with some measured results which are presented in chapter 6.

CHAPTER 2

System Design and Description

2.0 Introduction

Chapter 1 was provided to give a brief introduction to the overall disk drive channel proposed at Berkeley. This chapter will concentrate more on the system design of the digital adaptive equalizer which follows the Analog to Digital Converter found on the Uehara chip. The equalizer which is essentially an adaptive finite impulse response filter targets a $1\text{-}D^2$ partial response signal. There fore, we will begin with a review of partial response signaling and why it is convenient for disk drive channels. Ptolemy, a system simulator developed at Berkeley, was used to model the Class IV partial response channel and will be described. A brief presentation will then be given on adaptive equalization. Mainly the Stochastic Gradient algorithm was used to update the filter coefficients. Two update methods were studied, the Least Mean Squared (LMS) algorithm and the sign LMS algorithm, the relative trade-offs between these two methods will be studied. To achieve the aggressive goals for both high speed and low power, pipelining and parallelism were utilized for the filter architecture both of which have implications on the system stability. Therefore, all system non-idealities were studied and will be presented in this chapter.

2.1 Magnetic Disk Drive Channels

Before presenting the equalizer design we must first develop a channel model that properly characterizes the present day disk drives. The channel is best understood by first considering the physical operation of the disk drive.

Magnetic disk drive channels consists of three primary components, the write coil, magnetic media and the read coil. Information is conveyed to the disk by applying a current to a write driver which is being passed over a moving magnetic media. A magnetic dipole is then produced on the disk which has a direct correspondence to the direction of the current through the write driver. To read information off the disk a second coil is passed over the media. This coil will produce a current which is proportional to the change in magnetic flux through the inductive head. Therefore, as the coil passes over a region where the magnetic dipoles are changing direction a voltage pulse will be produced across the read head see Figure 2.1. Thus, it is important to note that a read head can only sense a change in the direction of stored information and not the actual information stored on the disk.

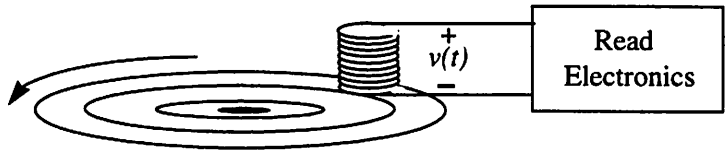


Figure 2.1 Read and write operation in disk drive channels.

2.1.1 Lorentz Pulse

Under ideal conditions the pulse produced at the read head would be an impulse spike. However, two physical limitations create an upper bound on the bandwidth(BW) of the read channel. The first has to due with the fact that the change in magnetic flux is not instantaneous as the read head passes over the media. Therefore, instead of the ideal voltage spike we receive a smeared pulse (Figure 2.1). A second contribution to the finite BW channel relates to the low-pass capacitive nature of the read electronics. Assuming the channel is linear, we can obtain a model which characterizes the continuous time impulse response by observing an isolated transition. The

Lorentz pulse is one such model that characterizes an isolated transition. The equation for the Lorentz pulse (shown in Figure 2.1) below relates the 50% point on the pulse to the symbol period. The parameter PW_{50}/T is now a relative measure of intersymbol interference in the channel. A high value of PW_{50}/T implies less distance in time between adjacent pulses (or symbols) resulting in more intersymbol interference. After consultation with manufactures of disk drive electronics (Quatum, National Semiconductor, and Silicon Systems) it was determined that a reasonable values for PW_{50}/T for modern drives ranges from 1.5 to 2.5 and will be assumed for all simulations presented later in this chapter.

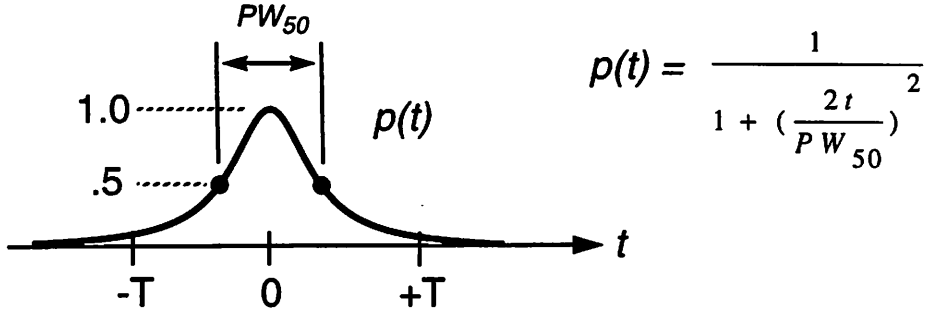


Figure 2.2 Lorentz pulse which relates the 50% point on the pulse to the symbol period.

$$p(t) = \frac{1}{1 + \left(\frac{2t}{PW_{50}}\right)^2} \tag{2.1}$$

Other models for the channel which account for the non-linearities associated with present day inductive heads include the Gaussian pulse, and the raised cosine function. However, the Lorentzian pulse is the most commonly used throughout the literature and will be used in this paper and in simulations. It is further noted that newer magneto-resistive heads are being developed which are shown to exhibit an extremely linear behavior.

2.1.2 Channel Response

Under the assumption that the channel is Linear Time Invariant we can write the received signal as a superposition of the current received symbol pulse and the interfering effect of the adjacent pulses. This is shown below where p(t) is the Lorentzian pulse and A_k represents the direction of a transition on the magnetic media. A_k is also referred to as the channel symbol.

$$v(t) = \sum_{k=-\infty}^{\infty} A_k \cdot p(t - kT) \tag{2.2}$$

From Eq. 2.1 and Eq. 2.2 we can now see the effect of superpositioning a series of adjacent transitions using the lorentz pulse in Eq. 2.1 (Figure 2.1)

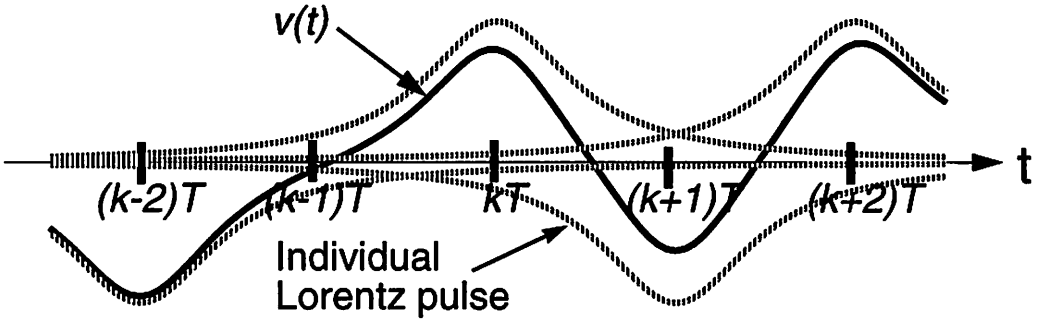


Figure 2.3 Effect of Intersymbol Interference on channel symbols.

Originally, there were two main methods for storing information on magnetic media: Non Return to Zero (NRZ) and Non Return to Zero Inverse (NRZI) [2]. In NRZ recording, information is represented on the disk by the direction of a magnetic dipole. Therefore, the sign of the write current is dependent on whether a one or zero is being written. Then when reading the information off the disk a transition in the dipole position corresponds to a pulse on the read head. A plus on the coil is interpreted as a change in binary information. In NRZI recording, a binary one is represented as a change in the direction of the magnetic flux where a binary zero is interpreted as a lack of change in magnetic dipole direction.

NRZ is disadvantageous in how the information is read back from the media[2]. As explained before a change from a 1 to a 0 is represented by a transition pulse. Therefore, if in the process of decoding the read message a peak goes undetected an error will be generated. However, the next peak to be detected will be interpret as a change from a 1 to 0 when it should have represented a transition from a 0 to a 1. Thus, the NRZ system suffers from an error propagation mechanism in the readback process. In NRZI recording a one bit is represent by a transition and the lack of symbolizes a zero. When a peak goes undetected this generates a single error and will not be propa-

gated through the system. Therefore, NRZI systems are preferred to NRZ and will be assumed for the remainder of this chapter.

To keep a constant symbol rate and radial velocity the spacing between dipole transitions on the inner diameter need to be spaced closer than the outer diameter. This change in density creates a variation in the channel characteristics from the inner diameter to outer diameter. This again can be specified by the PW_{50}/T parameter in the Lorentz pulse. Corresponding to earlier information from it will be assumed that the PW_{50}/T varies from 2.5 on the inner diameter to 1.5 on the outer diameter.

2.2 Partial Response

Partial Response signaling is a coding and decoding technique used to control the spectrum of the channel signal. The general from of the channel characteristics can be evaluated as a filter with a response of the form[11]&[6]

$$F(D)) = \sum_{i=0}^{N} f_i \cdot D^i \tag{3.3}$$

The more general form of this expression which reveals the capability of the channel response to control the spectrum is written in the form of

$$F(D) = (1-D)^m (1+D)^n \tag{3.4}$$

where m=0 or n=0 but not both. Three commonly used forms are the dicode where m=1 and n=0, the doubinary where m=0 and n=1, and the modified doubinary where m=1 and n=1. Each of these can be seen to contribute zeros to the frequency spectrum which can be favorable for increasing the Bandwidth of the channel.

The partial response system consists of feeding a signal through a channel with a response of *F(D)*. Then at the receiver end, the signal is passed through an inverse filter to recover the original information. In the absence of noise the output of the system shown in Figure 2.4 will equal the input. However, if the channel response is of the form of Eq. 3.4 then we see that the receiver inverse filter will be IIR. This implies that we are susceptible to propagation of errors. Thus, these type channels are usual precede with a precoder which alleviates the problem of error propagation.

For magnetic Disk drives a precoder is used which performs a mod2 operation on the input. A sequence detector is then used in the receiver electronics which performs the inverse function of the precoder.

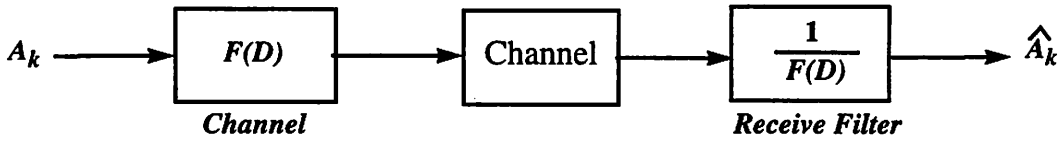


Figure 2.4 Coding and decoding filters usually found on partial response channels.

The question now arises of how we can apply partial response to a magnetic media and what are the advantages. We must first concentrate on which partial response polynomial will be of interest in magnetic disk drive channels. Recall that the read coil passing over the drive head is performing a differentiation on the information stored in the form of magnetic dipoles. Thus, by the physics of the read head we perform a 1-D operation on the data being passed through the channel. The next observation is that by increasing the density on the disk drive read head such that we deliberately create intersymbol interference and sample each isolated pulse twice we can create a 1+D operation on the information being passed through the channel (Figure 2.5) This leads to a channel response of (1+D)(1-D) which is the special case of the modified duobinary, often referred to as a class four partial response system (PRIV). An example of which is shown in Figure 2.6.

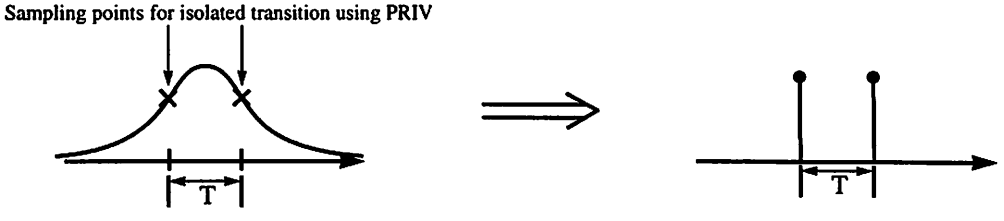


Figure 2.5 Sampling of a PRIV isolated pulse

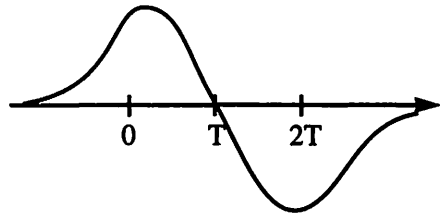


Figure 2.6 Isolated Pulse Response for PRIV

The key advantage of partial response signaling lies in the 1+D operation performed on an isolated pulse. This is better understood if we first consider the ideal isolated pulse for a peak detection system. Here, the optimal situation occurs when we equalize the pulse such that there is no contribution to the samples nearest neighbor, or in other terms the sampling rate on equalized pulses is such that there is no intersymbol interference. Now label the channel response of the peak detection system due to the read electronics only as h(t). Next, assume that we take the same read head with the same response h(t) and apply it to a class four partial response system. Here, the objective is to sample an isolated pulse such that we receive two symbols for each transition (see Figure 2.6).

Thus, the partial response system must have a sampling rate that is 50% faster than the peak detection system. This implies that the sampling period of the partial response drive is one third less than the peak detection system. We can write,

$$T_{PRIV} = \frac{2T_{peak-detector}}{3} \tag{3.5}$$

Thus, the main advantage of using a class four partial(PRIV) response system is that the spacing between adjacent transitions have now decreased by 33% achieving a net gain in bit density on the disk equal to 50%.

As mentioned before partial response systems need some form of data coding to prevent the propagation of errors. Therefore, the read electronics needs a method to decode the information. This is usually done in the form of a Viterbi sequence detector. Therefore, partial response systems have the extra complexity of more hardware over traditional peak detectors.

2.2.1 Simulation of PRIV disk drive channel

To perform later studies on the disk drive channel with equalization, ptolemy, a system simulator developed at Berkeley was used to model and create channel waveforms. Figure 2.7 illustrates the method used for modeling the channel. A Pseudo-Random number generator supplies a binary sequence A_k (1, 0) which is then passed through a Class IV partial Response Coder to create a three signal constellation B_k (+1, 0, -1). The B_k sequence is then convolved with a lorentz pulse

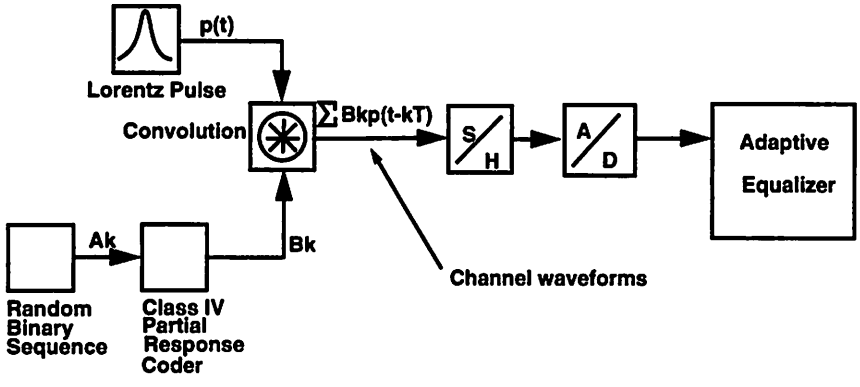


Figure 2.7 Channel model used in Ptolemy

which results in a linear channel signal. The channel waveform is sampled and passed through a 6bit quantizer to simulate the sample and hold along with the A/D operation done on the Uehara front end chip. The simulation samples found after the Analog to Digital converter were then stored in a file to be used as input for the adaptive equalizer in later simulations. To simulate variations in channel conditions the PW50/T parameter found in the Lorentz Pulse facet was varied from 1.5 to 2.5 per the discussion in section 2.1.1. Also, simulations were run with as much as

15dB of noise added to the channel signal. An example PRIV channel waveform with $PW_{50}/T=2.5$ has been shown in Figure 2.8.

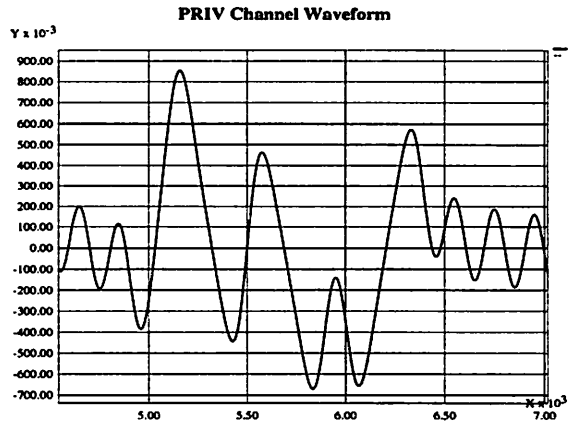


Figure 2.8 Example channel waveform produced by Ptolemy.

2.3 Adaptive Equalization

Recall that the ideal transmission of a symbol through the disk drive channel is provided in the form of a chain of impulse spikes separated in time by the sample period *T*. However, due to the non-idealities introduced by the disk drive electronics a low pass characteristic is displayed by the channel thus, creating a superposition of pulses which interfere with one another creating a phenomena know as Intersymbol Interference. To mitigate the low pass nature of the channel and recover data with a reasonable bit error rate a filter which provides a high frequency boost is usually required. For the specific case of disk drive electronics the equalization can be provided in either the analog or digital domain. Uehara [16] claims in his thesis that for analog signals suffering from ISI there exist an optimal when some channel equalization is provided before the analog to digital operation immediately followed by a further digital equalization.

Many channels experience a variation in characteristics, examples included modems and radios. In magnetic storage channels, a variation in the properties exist due to change in the spacing between symbols from the inner diameter to the outer diameter of the disk. Further variations in channel characteristics can be attributed to a change in the distance between the media and read head electronics. To provide equalization under varying channel conditions it is desired to have a

filter which can adapt to the changing channel conditions thus the term adaptive equalization. The focus in this project is to design an adaptive equalizer which performs a filtering function for varying channel conditions.

To implement our equalizer the stochastic gradient algorithm, also referred to as least mean squared algorithm (LMS) was used, the derivation of which is left to [13]. In the LMS algorithm coefficients are updated by passing the output of a finite impulse response filter through a slicer where a hard decision is made on the filter output. The slicer decision and the filter output are then used to generate an error signal. This error signal is then feedback and used in the LMS algorithm to update each coefficient.

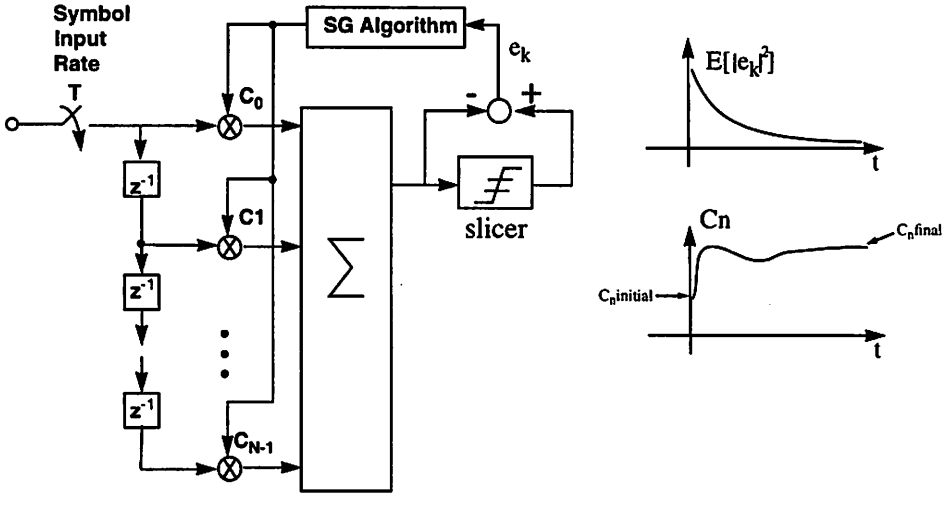
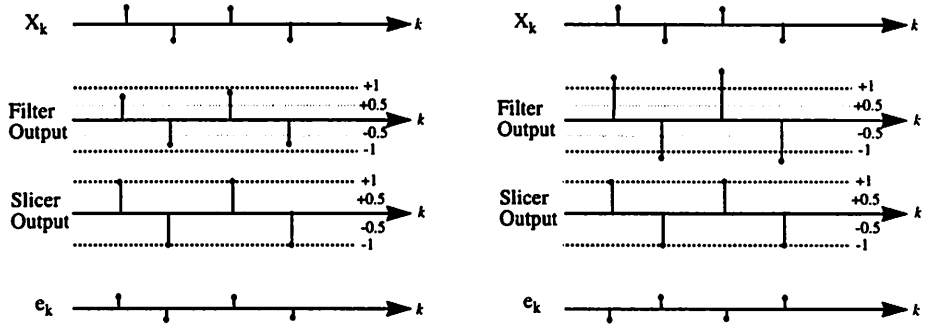


Figure 2.9 Adaptive Equalizer utilizing the Stochastic Gradient Algorithm.

$$C_{n,k+1} = C_{n,k} + \beta \cdot X_{k-n} \cdot e_k \tag{2.6}$$

From Eq. 2.6 we can gain some intuition into how the LMS algorithm adapts to varying channel condition. Each nth coefficient (Cn) is updated at time period k+1 using information based on the previous coefficient value (Cn,k), the sample aligned with the Cn (Xk-n) and the Error generated at the output of the slicer at time k denoted as e_k. β is a scale factor which helps control the magnitude of change from one coefficient in time to the next. Essentially the new coefficient is

being driven down a gradient to reduce the Mean Squared Error (denoted as $E[|e_k|^2]$) at the output of the slicer. A couple of examples may prove more intuitive.



(a) Filter with insufficient gain.

(b) Filter with excess gain.

Figure 2.10 Examples of Adaptive Equalizers with a gain error.

Assume we have three points in our signal constellation normalized to an amplitude of one (+1,0,-1). Also, the slicer is set to thresholds of +/-0.5. Figure 2.10a illustrates an example of when the filter is not supplying enough energy to the incoming samples and thus the amplitude of the output is below a desired value. From Figure 2.10a we can see that Beta, the error, and the aligned sample for a couple of points in time. Note the sign of the product of the $\beta \cdot X_{k-n} \cdot E_k$ always works in a direction so as to add more to the magnitude of the coefficients providing more equalization and in turn increasing the amplitude of the output samples. Further note that Eq. 2.6 acts to reduce the average power of the error presented at the output of the slicer, often referred to as the Mean Squared Error. The coefficients move along a gradient toward a final target value at which time the error becomes extremely small and inturn $\beta \cdot X_{k-n} \cdot E_k$ approaches zero and the coefficients remain relatively constant. If we assume that the feedback loop is stable the coefficients will remain relatively constant about their final target value until the channel conditions again change. As a second example, from Figure 2.10b we can see that when the filter output is too large the error signal generated has the effect of reducing the filter coefficients until the mean squared error at the output approaches zero.

Two factors influence the stability of the feedback network, the size of β, and the latency introduced from feeding back the error signal. Like any negative feedback network if the signal

being feedback is too large the system can go unstable. Similarly, a large delay in generating the error signal contributes to a degradation in the loop stability. A relationship also exits between the size of Beta and the delay in the feedback network, the more delay the smaller Beta must be to attain stability. Conversely, relatively short delays in the feedback path allow for a larger Beta inturn permitting faster convergence for the filter coefficients. A more detailed analysis studying the relationship between Beta and the delay will be outlined in the next section on the equalizer design.

2.4 Equalizer System Design and Simulation

An outline will now be given for the design procedure used to specify the system for our digital adaptive equalizer. System level simulations were run to study different architectural approaches to implementing the equalizer. Initially, spice simulations were run to determine exactly how fast the multipliers were able to run. It was determined that to achieve an effective 100MHz throughput it would be necessary to run four filters in parallel. In addition, to conserve area it was also desired to update the coefficients using the output from one filter only. Thus, in summary the system level simulations were run to study the stability of the adaptive process in the filters with all non-idealities incorporated into the simulations. It may be difficult to understand some of the non-idealities listed. If things appear unclear it may be beneficial to skip to chapter 3.0 where a detailed explanation of the adaptive equalizer architecture is given. The following is a list of the non-idealities studied through simulation.

- The effect of latency in the feedback path on the filter stability.
- The effects of quantization error in the coefficients and the sampled data on filter stability.
- The effect of not sampling the error every period. For example in our proposed architecture it would be convenient to use only one of every fourth error samples to generate new coefficients.

2.4.1 Update Algorithm and stability

Recall that in Eq. 2.6, that a trade-off exists between the size of Beta and the delay in the feedback network. To gain a better understanding of the relationship between Beta and feedback delay a closer analysis was made of Eq. 2.6. In particular, if we consider the update of each coefficient as being dependent on the previous coefficient value. We will make a further assumption that the

input samples remain relatively constant giving rise to a system similar to what is shown Figure 2.11.

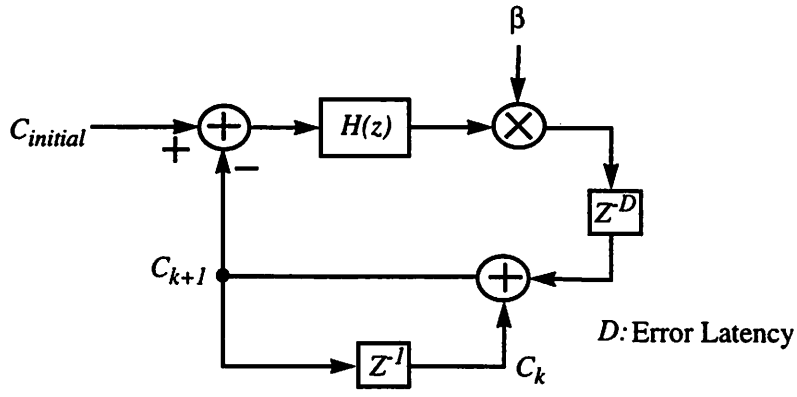


Figure 2.11 Simplified model for looking at the stability of an individual coefficient.

Figure 2.11 illustrates a simplified model for analyzing one coefficient. Basically, an initial c(n) is feed into a system h(n) which we assume is similar to the operation of generating the filter error. Now in Figure 2.11 we model the slicing operation as something similar to subtracting a desired coefficient value and in turn producing an error term. The error is then delayed by D periods and scaled by a factor of beta. It is worth mentioning that the system shown in Figure 2.12 is an attempt to model just one coefficient. However, as the number of coefficients in the system grows so does the relative instability. Therefore, the results of any simulations based on one coefficient will reflect an aggressive design for the stability of the filter coefficients.

The model illustrated in Figure 2.11 was implemented in Ptolemy and simulations were run for varying delays (D) in the feedback loop the results of which are shown in figure 2.12

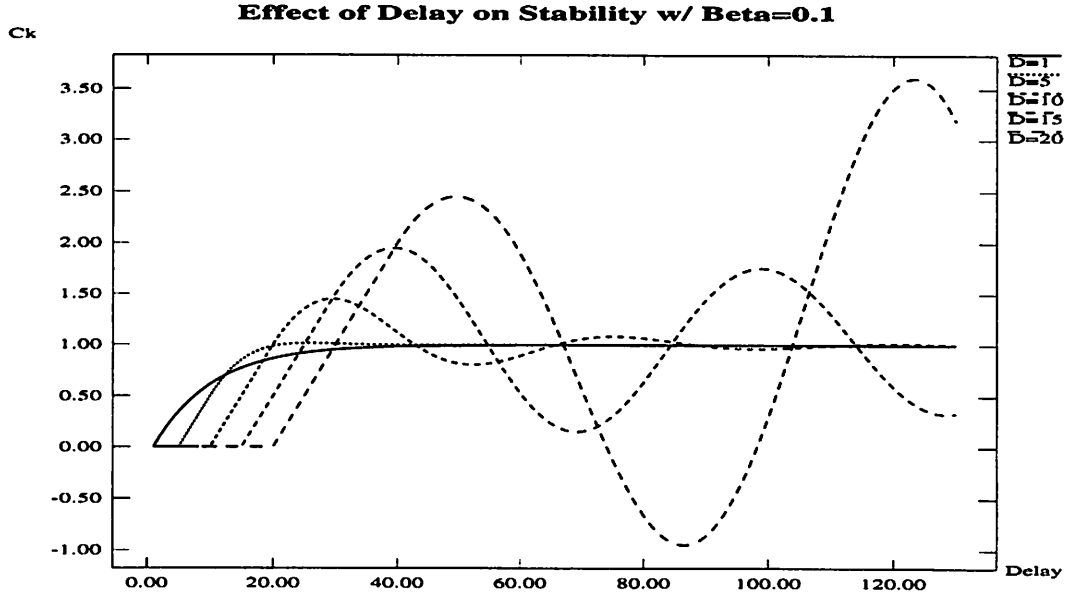


Figure 2.12 Simulation with varying delay of the simplified single coefficient model using the LMS algorithm.

From spice simulations of the proposed update circuitry it was determined that the filter coefficients would require at least 16 periods for $\beta \cdot X_{k-n} \cdot E_k$ to become available. As we can see from Figure 2.12 a delay of 16 leads to an unstable system for relatively small values of Beta. Therefore, at this point it was proposed to use a slight modification to the LMS algorithm known as the sign LMS algorithm. Using this algorithm it was determined that the latency in the feedback path could be reduced to 8 sampling periods.

$$C_{n,k+1} = C_{n,k} + \beta \cdot \text{sgn}\,(X_{n,k-n}) \cdot \text{sgn}\,(e_k)\,\text{r}$$

Essentially the signed based LMS algorithm only looks at the sign of both the error and the sample aligned to that error, β is then added or subtracted to the previous coefficient based on the results of $\text{sgn}\,(X_{n,k-n}) \cdot \text{sgn}\,(e_k)$. It was determined that by using this algorithm the delay in the feedback path could be reduced to 8 sampling periods due to the elimination of a multiplying operation required to produce $X_{k-n} \cdot E_k$ in the straight LMS algorithm.

Again a small simulation model was made for the update of an individual filter coefficient using the Sign LMS algorithm. Simulations results are shown in Figure 2.13 with varying delay in the feedback loop.

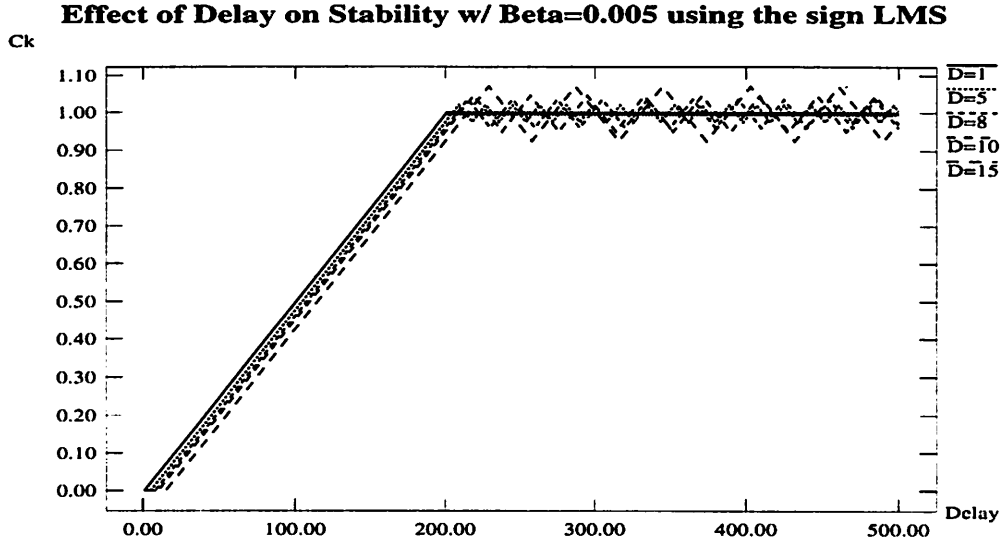


Figure 2.13 Results of the individual coefficient model for the sign LMS algorithm.

We can see that the coefficient value is stable for delay of 8 sampling periods in the feedback loop. However, note that for a larger Beta the final coefficient value becomes more course. The key observation is that the coefficient will converge for larger delays and has an acceptable settling accuracy for a feedback latency of 8. Therefore, the decision was made to implement the sign based LMS algorithm to update the filter coefficients in our adaptive equalizer design.

2.4.2 Equalizer Simulation with all Non-Idealities.

With the decision to use the sign LMS algorithm instead of the straight LMS further considerations were made for the entire design. To be compatible with the Uehara front-end chip the input samples were fixed to six bits. Also, it was desired to have a regular multiplier structure in the FIR filters, therefore, the coefficients were also fixed to six bits allowing each multiplier to be 6x6. However, to maximize the information about the previous coefficient 11bits were used to store the old coefficient to be used in the update processes with only 6 of them being feed to the input of the

filters. Again, due to reasons related to the hardware design (see chapter 3.0 for details) the coefficients were updated only once every fourth period. Simulations were run using eight taps for the filter.

2.4.2.1 Simulation Conditions

The main objective of the simulations was to check for convergence of the filter coefficients. This was done by observing that the filter coefficients converged on a value and are held constant for steady channel characteristics.

To increase the likelihood that the filter coefficients will converge a reasonable starting point is required. Therefore, initial coefficients were calculated and used for all simulations. A sample calculation of initial coefficient values is given in Appendix A.

The proposed Adaptive Equalizer utilizing the sign LMS algorithm was built in Ptolemy. All non-idealities related to the hardware were implemented in the simulation including the quantization effects caused by the coefficients, channel samples, and the finite word size of Beta. Simulations were also run with noise added to the input to reduce the SNR as low as 18dB. For a given set of initial coefficients the channel conditions were varied from a $PW_{50}/T=1.5$ to a $PW_{50}/T = 2.5$. The filter non-idealities are summarized below. A block diagram of the adaptive equalizer model used in ptolemy is shown in Figure 2.14.

- The error signal is only generated using filter 1. Thus, the coefficients are only updated once every four periods.
- The input signal is quantized to six bits as are the coefficients going to the four filters.
- 11 bits are used to store the previous coefficient value used in the update operation.
- Because of the delay incurred from the hardware, the error now has a latency of 8 sampling periods.
- The sign LMS algorithm is now being used instead of the traditional LMS algorithm which requires more computation to update the error, thus contributing to are greater latency in feeding back the error signal.
- Because of pad limitation on the final chip it was determined that only six bits could be used for Beta.

- Again, because of a Pad limitation only the three largest magnitude initial coefficients were supplied from off-chip inputs while the other initial coefficients were hardwired to ground.

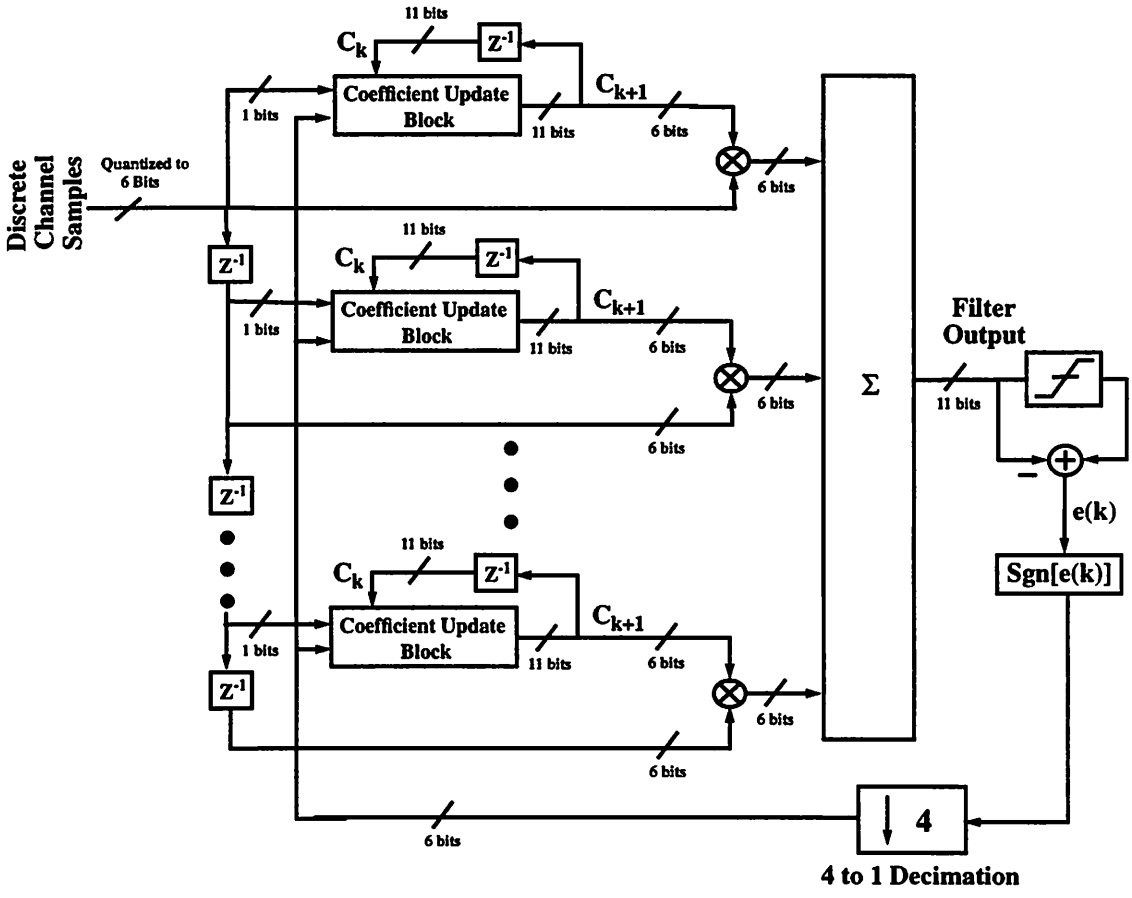


Figure 2.14 Ptolemy model of the adaptive equalizer used for simulation of the proposed architecture.

2.4.2.2 Simulations Results

Simulations did show that the filter converges and remains stable for the given design using the sign LMS algorithm. In particular, the reduced latency in feeding back the error signal resulted in a more stable system allowing the filter to converge for a wider range of Beta.

To illustrate the robustness of this filter design the worst case channel conditions (PW_{50}/T =2.5) were used for the purposes of illustration. The channel was simulated with an 18dB SNR at the input however, for purposes of illustration only simulations without noise applied to the input

are shown. From figure 2.15 we can see all eight of the filter coefficients adapting to a channel with a PW_{50}/T =2.5. In Figure 2.16 the error power has been plotted as a function of Beta. It is interesting to note that for smaller values of Beta the filter will more easily converge and once the filter coefficients have reached a final solution the error power will be less which implies the filter is

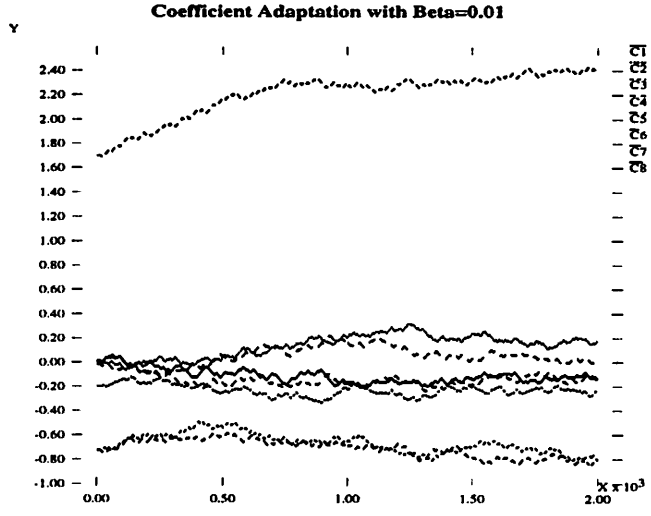


Figure 2.15 Filter coefficients adapting to a channel with a PW_{50}/T=2.5.

providing better equalization of the channel samples. However, as the value of Beta becomes

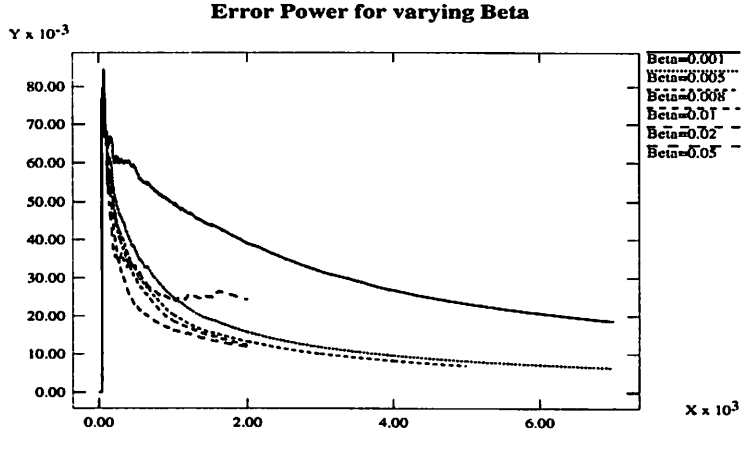


Figure 2.16 The Error power during adaptation for different values of Beta.

larger the filter has a harder time converging on a solution, the error power stays large and the channel samples are not provided with as much equalization.

It is also interesting to note the effect of the equalization on the channel symbols both before and after the filter has adapted. In figure 2.17 a plot is shown of the channel symbols at the input and output of the filter before the adaptation has begun. This can be contrasted with figure 2.18 which shows the filter output after the equalizer has had a chance to converge on the final values. From figure 2.18 we can see that the symbols at the output of the filter are either +1,0,-1 which is what we should expect.

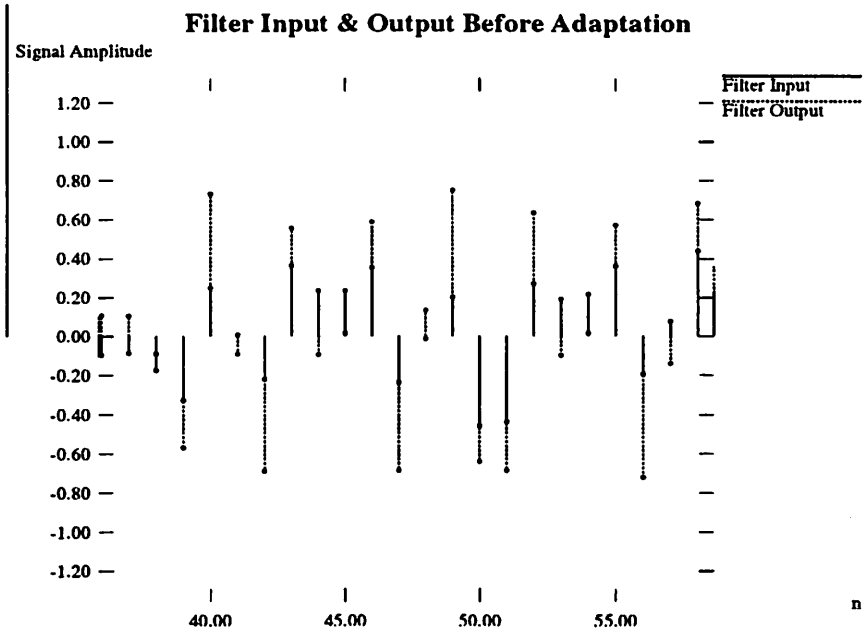


Figure 2.17 Channels samples passing through the Equalizer before the filter has adapted.

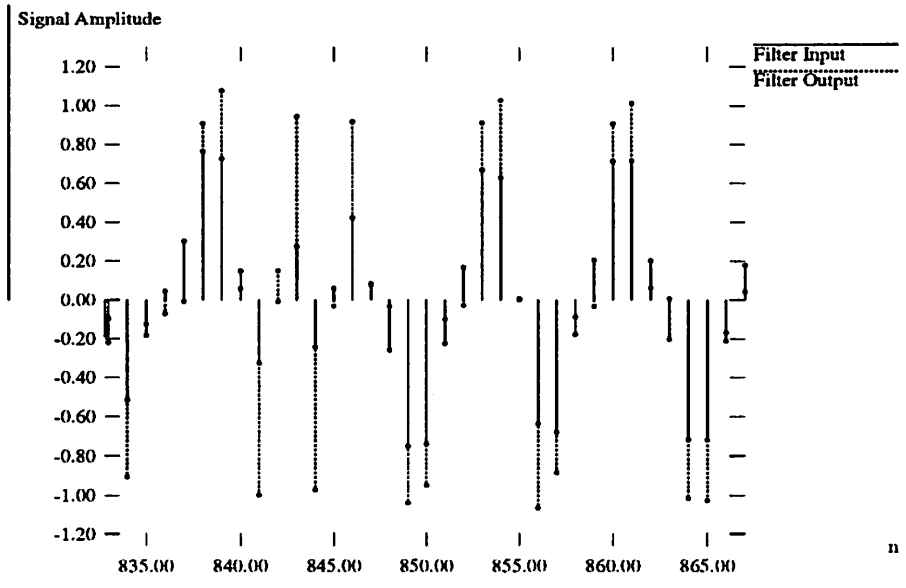


Figure 2.18 Channel samples after the filter has completely adapted.

2.5 Summary

In this chapter we have seen the design and verification of an eight tap adaptive equalizer. The use of the sign LMS algorithm was used to reduce the latency in the feedback path of the error signal. Also, only one of four error signals are used to update the filter coefficients. Ptolemy simulations were used to prove that the filter can converge and is stable for the proposed architecture. The following chapters will focus on how the proposed system was implemented into hardware.

CHAPTER 3

DSP Architecture

3.0 Introduction

Throughout the first chapter we saw a general description of the disk drive channel proposed at Berkeley, while chapter two gave the details surrounding the system design and specifications. Now, it will be desired to focus on the implementation of the system design of the adaptive equalizer. In particular, this chapter will give an overview of the general DSP architecture while chapter 4 presents the specifics behind the circuit design for the coefficient update.

To gain an understanding into the aspects of the DSP architecture we must first reflect on the original goals of this chip. This chip is specified to perform equalization at 100Mbits/sec on a magnetic disk drive channel which employs a Class IV partial response signaling. Also, an additional requirement is to perform all the equalization functions while consuming less than 400mWatts. To be compatible with current standards found in the semiconductor industry a 1.2um CMOS process was chosen for the circuit design and layout. The need to have both high speed and low power led to an architecture which employs both pipelining and parallelism, two technics commonly used to

increase the throughput while lowering the total consumed power by a specific function. Table 2.0 presents a summary of the main DSP specifications. 1

Table 1

	Specification
Data Rate	100Mbits/sec
Power	400mW <
Technology	1.2um CMOS
Supply	3.3 Volts

The following sections present an overview of the details surrounding the architectural implementation of the adaptive equalizer.

3.1 General Description

Three main components are necessary to perform digital adaptive equalization, a finite impulse response filter (FIR), a slicer at the output, and a block which computes adjusted values for the filter coefficients. From figure 3.0 we can see that the filter and coefficient update blocks are both very regular (repetitive blocks) lending itself well to the application of low power design technics such as pipeling and parallelism. It has been shown in Wong's Thesis [15] that filter speeds on the order of 100Mbits/sec are not possible in the selected 1.2um technology. Therefore, a decision was made to run four filters in parallel to meet the throughput requirement of a 100MHz. Wong's thesis further demonstrates that an overall power savings can be obtained when running four parallel filters at a fourth of the sampling rate.

The filter coefficients are updated using the stochastic gradient algorithm. As specified in the chapter 2, the sign LMS algorithm was used to improve the stability of the feedback network. Through system level simulations it was further learned that only the output from filter1 was necessary to perform the coefficient update. This lead to a reduction in both the hardware and com-

plexity associated with implementing the coefficient update block. From Figure 3.1 we now get a more accurate representation of the actual system used to implement the adaptive equalizer.

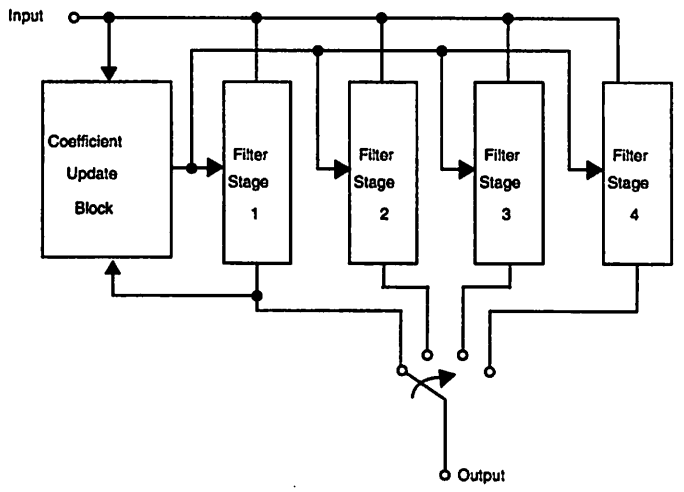


Figure 3.1 Four Parallel filters with only filter1 updating the coefficients.

3.1.1 Clocking

Synchronization for the entire chip is accomplished using four phases of a 25MHz clock which is derived from a 100MHz clock inputted to the chip. New data to filter1 and the operation of filter1 commence on the falling edge of clock1. Likewise, samples and the coefficient values are passed to the second filter on the falling edge of clock2. Filter3 and filter4 are clocked using clock3 and clock4 respectively. Operations in the coefficient update blocks are coordinated using clock1. A timing diagram may be found in.

3.1.2 Input Latches

To facilitate testing and clock generation the sampled data was inputted to the chip via two parallel channels. For example, if the sampling rate is 100MHz then two parallel channels with data running at 50MHz was inputted to the chip, however, the effective throughput of data coming in and leaving the chip is still 100MHz. Corresponding to the two input channels dual delay lines were run along the input to filter1.

3.1.3 Viterbi Sequence Detector

Two half rate Viterbi Sequence detectors were used at the output of all four filters. The class IV partial response polynomial lends itself well to the application of sequence detection. Recall that the incoming samples to the filter are coded with a $1\text{-}D^2$ response. Therefore, to decode the equalized output of each filter a sequence detector has been placed at the output of filters1&2 with another at the output of filters2&4.

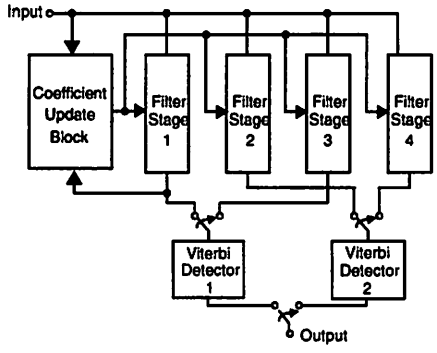


Figure 3.2 Four Parallel filters with the Viterbi Sequence detector.

From Figure 3.2 we can see that the viterbi decoders are arranged at the output so as to accept ever other sample being supplied from the equalizer. The Viterbi decoder then calculates the minimum path metric for a $1\text{-}D^2$ response. Further information about the layout and design of this block may be found in the thesises by Wong[15], Uehara [16].

In section 3.1, a very high level description of the key adaptive equalizer blocks was given. Now it is desired to give a more detail description of how the key blocks interact and the flow of data. The author of this Thesis was the designer of both the system and the coefficient update blocks. However, the design and layout for the filters was done by C. S. Wong. Therefore, only a brief description will be given for this block in the context of its interaction with the coefficient update blocks.

3.1.4 Operation of parallel filters

To begin we assume that the data has passed through the front end Analog to Digital converter illustrated in Uehara's thesis[16]. The input is supplied via two parallel channels as described

before. Let us first focus on how data is passed through the filter, refer to Figure 3.3. First assume that the adaptive equalizer has been turned off and we suddenly power up the device.

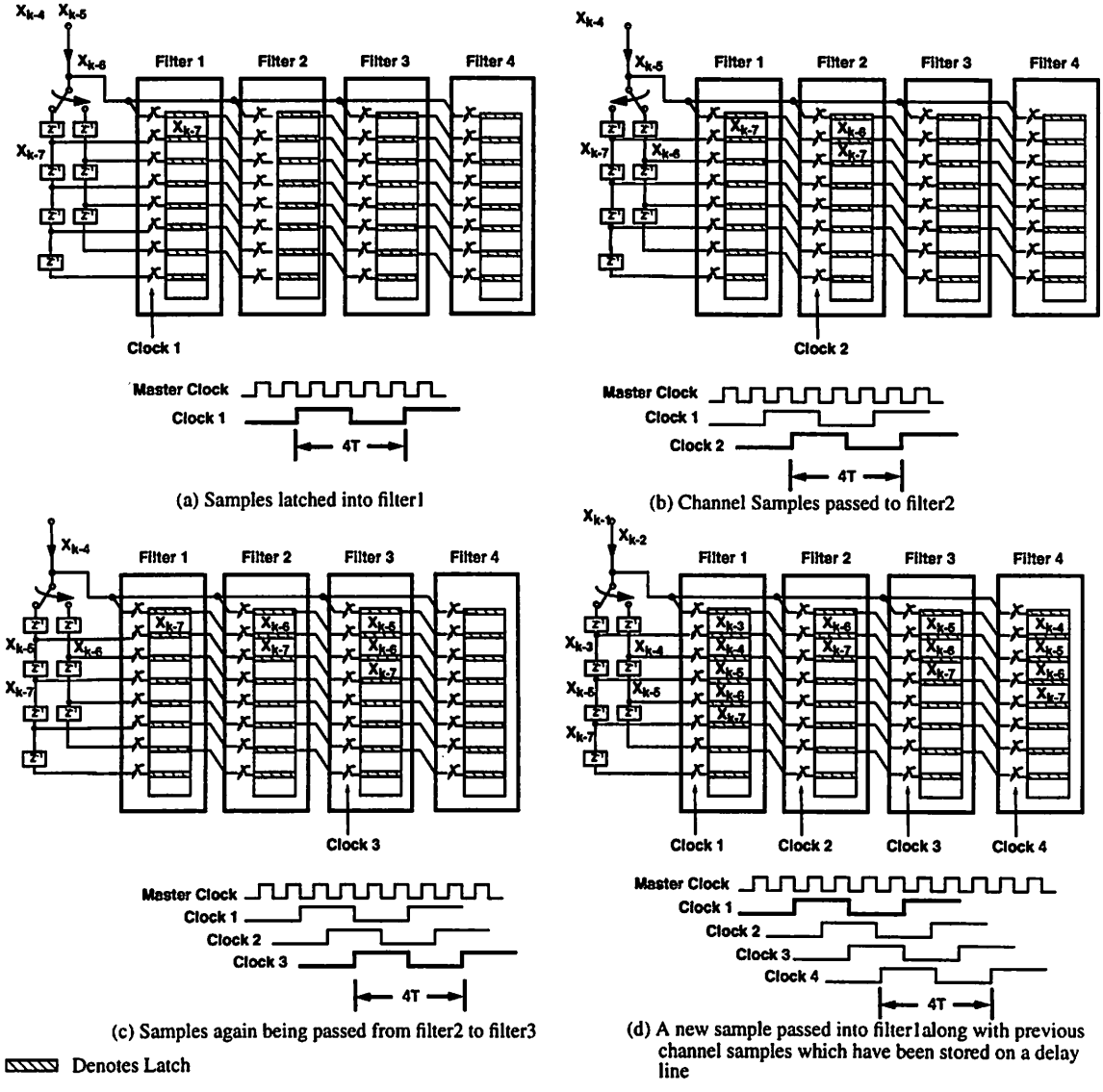


Figure 3.3 Data flow through the filter blocks.

assume that the very first sample to enter filter1 is X_{k-7} with all other data in the delay line and the other three filters equal to zero. Now on the falling edge of clock1 the input sample and the coefficients are latched into filter1 and the multiplication process begins (Refer to Figure 3.3a). At this time, the data latched into filter1 is immediately passed to the input of filter2. On the falling edge of clock2 X_{k-6} is latched into filter2 with the other seven inputs coming from filter1 (Figure 3.3b).

Filter2 then begins the multiplication process while passing all the of its sampled data to the input of filter3. Now when clock3 goes high X_{k-5} is the first sample latched into filter3 with the other seven samples coming from filter2. Once again this processes is repeated for filter4 when clock4 rises and X_{k-4} is available (Figure 3.3b). It should also be noted that X_{k-6} through X_{k-4} are being stored in a delay line as shown in Figure 3.3 at the input of filter1. Now when clock1 again goes high the samples stored on the delay line and a new sample X_{k-3} are latched into the input of filter1(Figure 3.3d). The whole process of passing data from one filter to the next is again repeated.

Recall that all four of the filters take eight clock periods from the time an input is applied to the filter until the output is ready. Therefore, in our example, X_{k-6} is applied to filter1 on the first period of clock1. During this period of clock1 the samples are multiplied with the coefficients. Now on the next period of clock1 the results of the multiplication are then feed to the accumulators and passed to the output. Thus, a total of eight clock periods are required for data to pass from the input of the filter to the output.

3.1.5 Coefficient Update

With an understanding of the filter blocks we will look at the architecture used to implement the sign LMS algorithm. Because only the first filter is used to update the coefficients for all four filters only the output from filter1 is passed through a slicer. The error signal generated from the slicer is then feed back to eight coefficient update blocks. Depending on the sign of the error a choice is made on whether to add or subtract beta from the previous coefficient value. The results of the selection in the coefficient update block are then applied to the input of filter1.

To fully understand the operation of the coefficient update block a step by step description of operation is given below. We will start the discussion of operation at the output of the multipliers in filter1. Recall that when clock1 goes high the outputs of each multiplier in filter1 are passed to the accumulators. Then during a single period of clock1 all the operations required to update the coefficients are performed. While working through the following explanation refer to Figure 3.4.

1) Clock1 goes high and the outputs of each multiplier in filter1 are latched into the accumulators. Also, when clock1 goes high both the value of the previous coefficient value plus beta and minus beta are immediately computed in each of the coefficient update blocks. Both the coefficient plus beta and minus beta are performed immediately after clock1 goes high. This dual computation is performed to help reduce the overall delay in the feedback path.

2) After the accumulators have finished summing the outputs of all eight multipliers the output of filter1 is then passed to a slicer. The slicer then makes a hard decision on the output of the first filter. Because we are using a class four partial response system with pulses normalized to 1 the output of the slicer will be either a +1, 0, or -1.

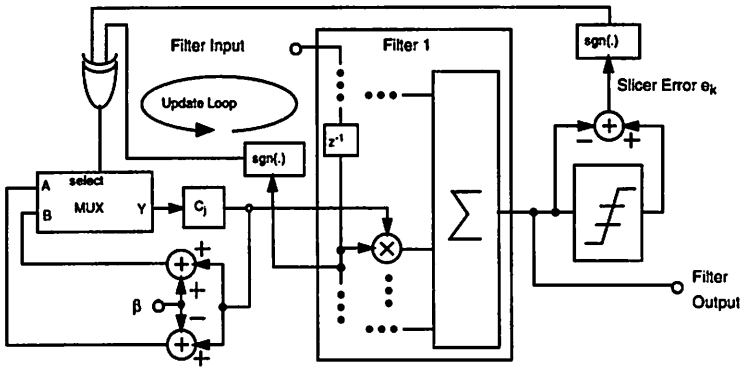


Figure 3.4 Sign LMS algorithm used to update the coefficients

3) The output of filter1 is then subtracted from the result generated from the slicer to produce the error. The sign of the error is then determined and feeded back to each of eight coefficient update blocks.

4) The sign of the error and the sign of each sample are inputted to an XOR gate. The output of the XOR gate is then used to select $C_k + \beta$ or $C_k - \beta$.

5) The new coefficients are now ready to be latched on the next rising edge of clock1 at the input of filter1.

6) Coefficients from the first filter are passed to filter2 on the rising edge of clock2. The same set of coefficients are subsequently passed to filter3 and filter4 on the rising edges of clock3 and clock4 respectively. We can see that the coefficients are passed from one filter to the next in a fashion similar to passing the channel samples between filters. However, to perform the proper operation of a finite impulse response filter the channel samples move to the right and down through the chain of filters unlike the coefficients which move strictly to the right through the parallel filters (refer to Figure 3.3 and Figure 3.5). This movement of data through the filter correctly implements the desired response of

$$y(n) = \sum_{k=0}^{M} Ck \cdot x(n-k)$$

where M = 7 for an eight tap finite impulse response filter.

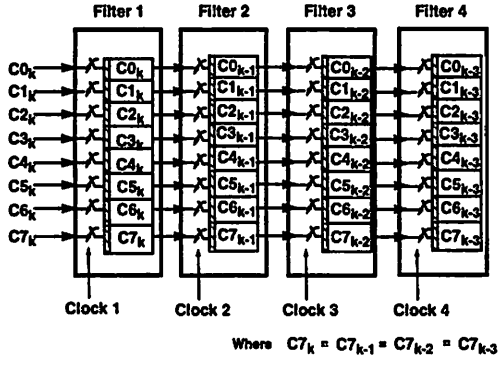


Figure 3.5 Movement of coefficients through all four filters.

Each coefficient update block is provided with the circuitry to both add and subtract the value of beta from the previous coefficient value. Also, the sign of the error is used rather than the actual error value. A combination of these the two mentioned features allow a rapid update of the coefficient values, thus providing more stability for the entire filter.

3.2 Chip layout and Organization

Because of the regularity associated with the design of both the filters and the coefficient update blocks the layout of the chip was produced in a fashion similar to what is shown in Figure

3.2. The chip is comprised of four parallel filters which mainly consists of eight 6 bit by 6 bit multipliers along with eleven accumulators. From Figure 3.6 we can see that the error generation circuitry has been shown on the bottom with all eight of the coefficient update blocks displayed on the left hand side of the chip. All four phases of the 25MHz clocks1 through 4 were placed at the bottom of the chip.

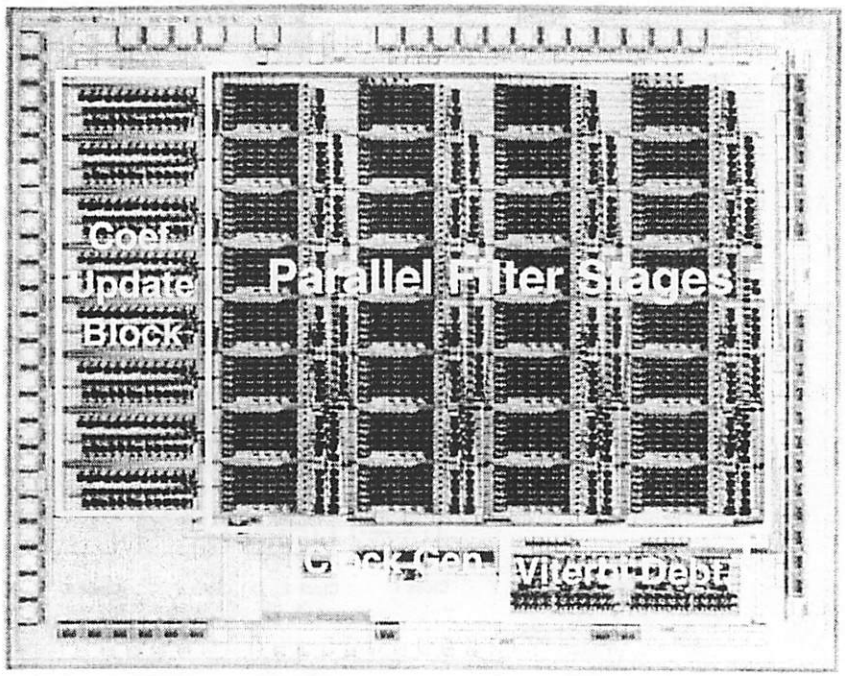


Figure 3.6 DSP Die Photo.

3.3 Summary

From this chapter we gained an understanding into the general layout and architecture of the DSP. Mainly, both the idea of pipelining and parallelism were used to increase the allowable effective sampling rate to 100MHz while reducing the overall power consumption. We also saw in this chapter that the sign LMS algorithm was used to reduce the latency in the feedback path. Layout of the chip was shown to be very regular due to the parallelism and the similar operations performed by both the filters and the update blocks.

CHAPTER 4

DSP Circuit Design

4.0 Introduction

Throughout chapters 2 and 3 we saw the design of both the system and the overall architecture associated with the digital adaptive equalizer. In this chapter we will focus on the specifics surrounding the circuit implement of the DSP chip, particularly the circuits associated with the coefficient update algorithm. For details on the design of the four filters and the viterbi sequence detector consult [15], and [16]. All the material described in this chapter will assume a knowledge of both the chip architecture and the algorithm used to update the filter coefficients each of which are described in chapters 2 and 3. Discussion of the coefficient update circuit design will begin at the output of filter1 and continue through the feedback loop to the point where the new coefficients are passed back to the input of filter1.

To perform all the coefficient update functions described in section 3.1.5 three main circuit blocks were design. First, the slicer, subtractor and the sign of the error function were combined into a single decoder. Basically, the output of filter1 is passed into a large decoder which produces the sign of the error, this signal is eventually feed to the eight coefficient update blocks. The sec-

ond main circuit block in the feedback process are the adders and subtracters used to calculate the previous coefficient value with a constant (Beta) either added or subtracted. A third custom design was performed for a multiplexing network which selects a new coefficient value based on the results from a computation done for the sign based LMS algorithm. For reason related to the stability of the entire filter, speed was the main issue in designing these circuits rather than power which was not perceived to be a problem because of the relative size of the update hardware in relationship to the overall size of the chip.

4.1 Timing requirements

Recall from chapter 3 that during one period of clock1, data latched at the output of the multipliers must be summed in the accumulators, pass through the slicer and subtractor after which the sign of the error is determined. This error signal in conjunction with the sign of the aligned sample X_k is then used to choose either $C_k + \beta$ or $C_k - \beta$, the new coefficient is then latched at the input of filter1. Further recall, that one period of clock1 is equivalent to four periods of the Master Clock. Therefore, if we are clocking the chip at 100MHz we know the period of the Master clock is 10nsec and there is only *40nsec to complete the accumulation along with all the update functions.* From the filter design it was known that the accumulators took approximately 2T of the Master clock, this leaves another 2T to slice the output pass it through the subtractor and select the previous coefficient with beta added or subtracted. Recall from chapter3 that the operation of adding or subtracting beta from the previous coefficient value is done in parallel with the accumulation, slicing and subtracting functions. Thus, both Ck + β and Ck - β are waiting for the sign of the error to be determined. We can now partition the 2T (20nsec) allocated for the update operation and identify the critical path.

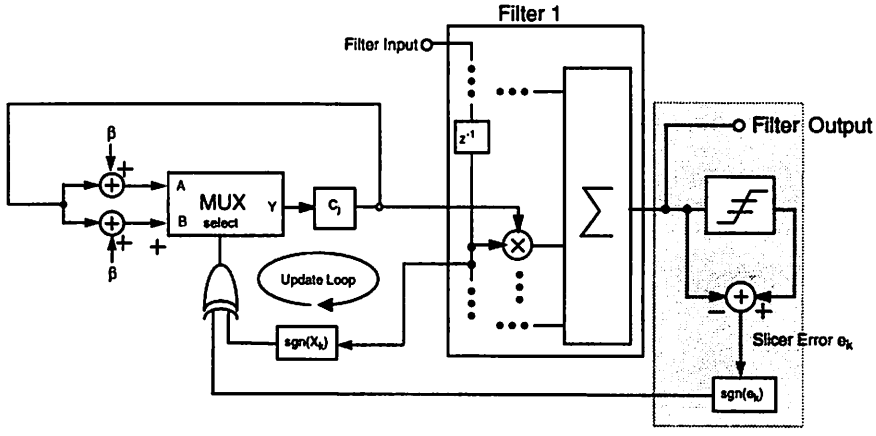
Once the sign of the error signal reaches the coefficient update block only a small amount of time is needed to multiplex the new coefficient values and route them to the input of filter1. Thus, a majority of the 2T allocated for the update operation can be dedicated to the slicing and sign of the error generation. For our design an initial goal of 1.4T (14nsec @ 100MHz) was devoted to slicing the output and generating the sign of the error signal leaving 0.6T (6nsec @ 100MHz) to route the new coefficients to the input of filter1. It must further be noted that this argument

assumes that the new coefficients are ready when the error signal arrives. Thus, the calculation of Ck + β and Ck - β must be completed in less than 1.4T. A summary of the timing requirements for each update function is listed in table 4.0.

Table 4.0

Circuit Function	Specified time (T)
Slice/Subtract/sgn(error)	1.4T
Calculate Ck + β & Ck - β	1.4T
Multiplex and route new coef.	0.6T

4.2 Slicer and Subtracter Design



The function of slicing the filter output, performing the subtraction to produce the error and obtaining the sign of the error were combined into one decoding block. Essentially the output of the first filter is feed into a decoder which immediately evaluates the sign of the error. This signal

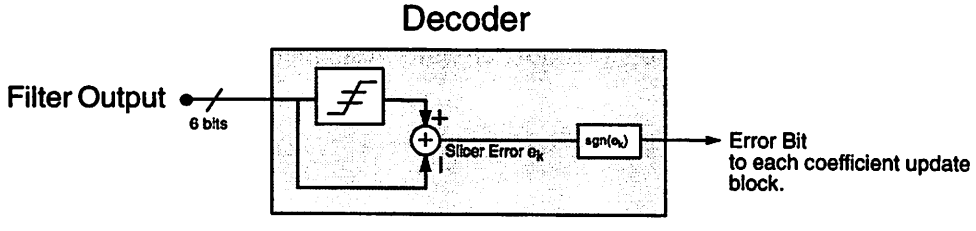


Figure 4.1 Decoder to implement slicing, subtracting and sign of the error generation.

is then feed to the inputs of all eight coefficient blocks. Before proceeding with the decoder design threshold levels for the slicer were selected and mapped to bit values.

4.2.1 Design of Slicer Threshold

To design the decoder it was first desired to determine the exact relationship between the bit values coming out of the filter and where to set the thresholds for the hard decisions coming from the slicer. Therefore, a truth table was made for the output of filter1 displaying the mapping between the six bit output and the channel symbol (for class IV partial response symbols, +1, 0, -1) along with the sign of the error produced. In appendix B we can see the described truth table. Threshold were selected in such a manor as to increase the available bit representation between +1 and -1 since this is where a vast majority of the channel symbols lie for class IV partial response due to the effects of Intersymbol Interference. However, sufficient room was also provided to ensure that an overflow problem would not occur. From appendix B, we can also see that carefully choosing the threshold can reduce the number of bits needed from the output. For example, assigning the symbol 1 to the two's compliment number +15 and the channel symbol -1 to be represented by -16, the slicer thresholds +/-0.5 now lie between 7 and 8 or -8 and -9 respectively. Notice that to implement this system only requires the three most significant bits from the output of the filter, thus reducing the number of gate delays required to decode the error signal and inturn the latency in the feedback path. However, now note that there is a considerable amount of headroom left above the symbol 1, that is in our example from 16 to 31 is now left as a protection against overflow while fewer bits are used to represent the symbols which lie between +1 and -1, thus degrading the SNR. Therefore, we can now see the trade-offs when scaling the output of all the filters. For a finite precision word, using large numbers to represent the symbol 1 implies less head room for overflow protection and more circuitry to implement the desired decoding function while

increasing the representation for the output samples between +1 and -1 which inturn improves the SNR at the output (refer to Figure 4.2).

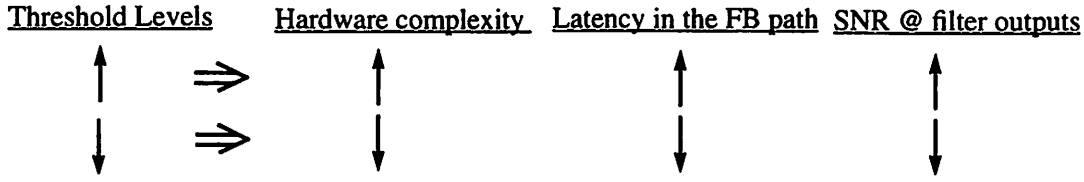


Figure 4.2 Trade-offs for threshold selection.

Ptolemy simulations were run to determine the maximum filter output. It was found that a sufficient protection against overflow is provided if the output of the filter is allowed to reach a symbol value of 1.3. Therefore, in designing our system the channel symbol 1.0 was scaled to the two's compliment number 24 with -1.0 represented by -24 (see appendix B). Now the Threshold levels of +/-0.5 correspond to 11 and -12 respectively. From appendix B we see both the symbol and the sign of the error can be decoded using only the four most significant bits from the filter output. It should be further noted that the sign of the *error was represented as a binary 1 for negative values and a 0 for positive error values.*

4.2.2 Decoder Design.

After a bit mapping was completed the next task was to design the logic necessary to perform the function of decoding the sign of the error from the six bit filter output. To aid in the decoder design a logic minimization program called expresso was used to reduce the truth table in appendix B to the minterms of the filter outputs A6, A5, A4, and A3. From expresso the following equation was obtained for the sign of the error denoted as *sgn(error)*

$$\text{sgn}(error) = \overline{A_6} \bullet A_4 \bullet \overline{A_3} + A_6 \bullet A_5 \bullet \overline{A_4} \bullet \overline{A_3} + \overline{A_6} \bullet \overline{A_5} \bullet \overline{A_4} + A_6 \bullet \overline{A_5} \bullet A_4 + \overline{A_6} \bullet A_5 \bullet A \quad (1)$$

Using DeMorgan's Theorem the Minterms were rearranged to obtain a solution for the *sgn(error)* using only NAND gates which requires two levels of logic and thus two gate delays. From equation2 we see that a four input NAND gate would be necessary to decode the sign of the error. However, through simulation it was shown that two NAND gates with a NOR gate can be made faster than a four input NAND gate.

$$\text{sgn}(error) = \overline{(\overline{(\overline{A_6}A_4\overline{A_3})} \bullet \overline{(A_6A_5\overline{A_4}\overline{A_3})} \bullet \overline{(\overline{A_6}\overline{A_5}\overline{A_4})} \bullet \overline{(A_6\overline{A_5}A_4)} \bullet \overline{(\overline{A_6}A_5A_4)})} \quad (2)$$

Therefore, Equation (2) was again rearranged so that only gates with three inputs or less were required. Equation 3 is the final form used to implement the sign of the error decoder.

$$\text{sgn}(error) = \overline{(\overline{(\overline{(\overline{A_6}A_4\overline{A_3})} \bullet \overline{(A_6A_5 + \overline{A_4}\overline{A_3})})} + \overline{(\overline{(\overline{A_6}\overline{A_5}\overline{A_4})} \bullet \overline{(A_6\overline{A_5}A_4)} \bullet \overline{(\overline{A_6}A_5A_4)})})} \quad (3)$$

The logic diagram is shown in Figure 4.3. Device sizes were selected to give the minimum progation delay through all layers of logic.

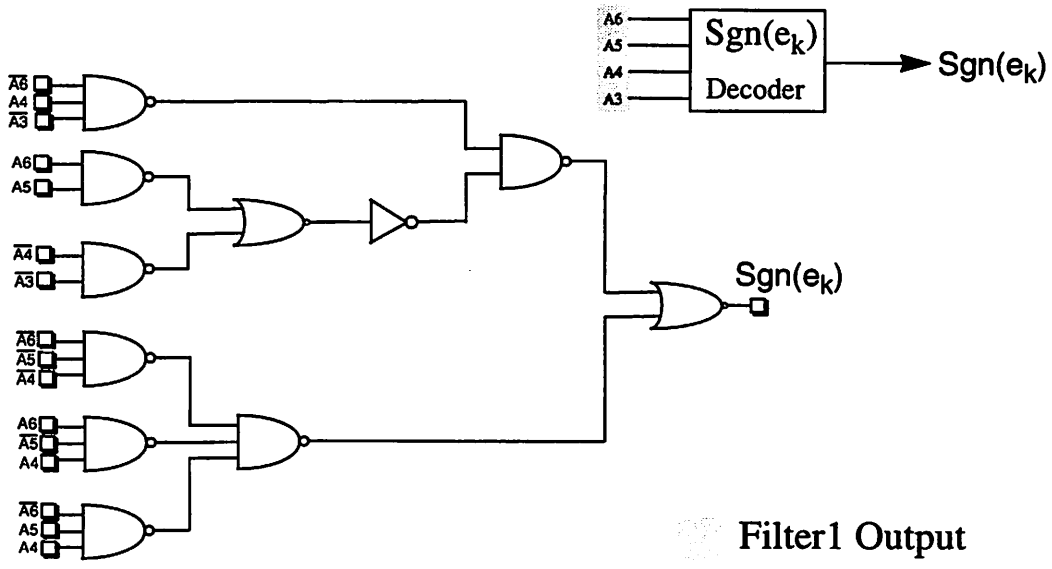


Figure 4.3 Sign of the Error Decoder.

Standard static CMOS logic was used to implement all the gates shown in Figure 4.3

4.2.3 Error Driver Design

Recall that the signal coming out of the decoder must now feed eight of the coefficient update blocks. At the input of each coefficient update block the sign of the error is passed through an XOR with the MSB from the corresponding sample. Thus, the output of the decoder must drive the inputs to eight XOR gates in addition to a large routing capacitance which runs completely across the chip (refer to Figure 4.4). From table 4.1 we know that the allotted time for the feedback signal to propagate from the output of filter1 to the input of each coefficient update block is 1.4T, or

14nsec at 100MHz. To attain this speed goal a custom driver was designed at the output of the decoding block.

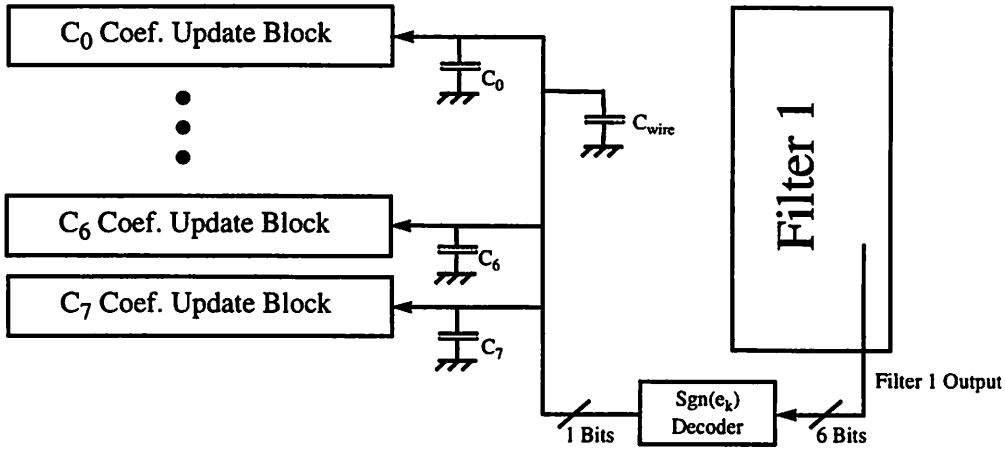


Figure 4.4 Parasitic Capacitance driven by the output of the decoder block.

To lower the delay induced by the capacitance at the decoder output several inverters were cascaded in a row. To determine the number of inverter stages necessary and the sizing of these buffers a design technique outlined in the reader by Jan Rabaey was used [17]. Rabaey has shown that there exists a relationship between the optimal numbers stages and the ratio of the loading capacitance to the input capacitance seen by a minimum sized inverter, this is repeated below in equation 4

$$N = ln(x) \tag{4.7}$$

Where N is the number of stages and x is the ratio of the loading capacitance to input capacitance of a minimum sized inverter. Applying the above equation to our situation for the loading capacitance seen by the output of the error decoder we have that $x = \frac{C_L}{C_i}$ where C_L is the total loading capacitance seen by the error decoder and C_i is the input capacitance for a minimum sized inverter used in the decoder block. For the 1.2um CMOS technology, it was determined that C_i = 14.69fF while C_L=1000fF resulting in x = 68.07. Plugging x into Eq. 4.7 we see that the optimal number of stages is 4.22. Four stages were used in spice with extracted capacitance values for C_L. However, through simulation it was observed that three stages were just as effective as four. Thus,

only three stages were used, Figure 4.5 shows the circuit diagram of the decoder buffer with actual transistor sizes.

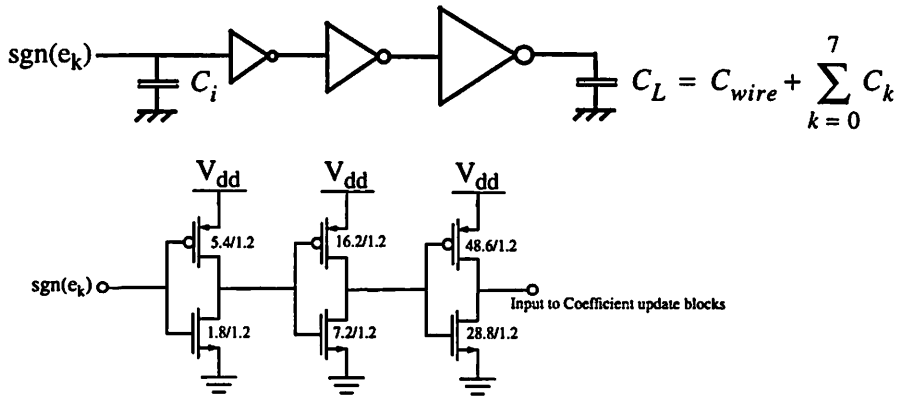
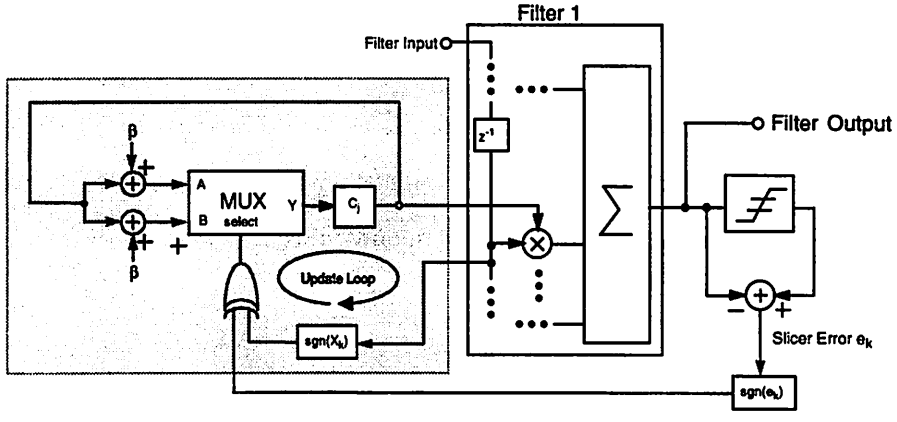


Figure 4.5 Buffer for $sgn(e_k)$

4.3 Coefficient update block design.



Each coefficient was supplied with a circuit block to update the corresponding tap value. The individual update blocks execute three main functions. First, a custom XOR gate was design to make a decision on whether to select Ck + β or Ck - β, based on the sign of the error and the sign of the corresponding sample both of which are inputted to the update block. A second function of the update block is to calculate both Ck + β or Ck - β every period of clock1. A third and final custom block designed for the update operation was a multiplexer used to select one of three possible values for a new coefficient.

From Figure 4.6 we see the general layout of this block. Inputs to the block are in the form of initial coefficients, the value of beta, and a control signal (Init) used to signify the initialization of the coefficients. Under normal operation the update block will begin calculating Ck + β and Ck - β on rising edge of clock1. An XOR gate is used to evaluate the sign of the error and the sign of the corresponding sample, the output of this gate then feeds a custom designed three to one mux which selects either Ck + β, Ck - β or a set of initial coefficients. The initialization of new coefficients during a start phase is performed by holding the INIT signal high which disables the signal coming from the XOR gate and allows the passage of new coefficients to the input of filter1.

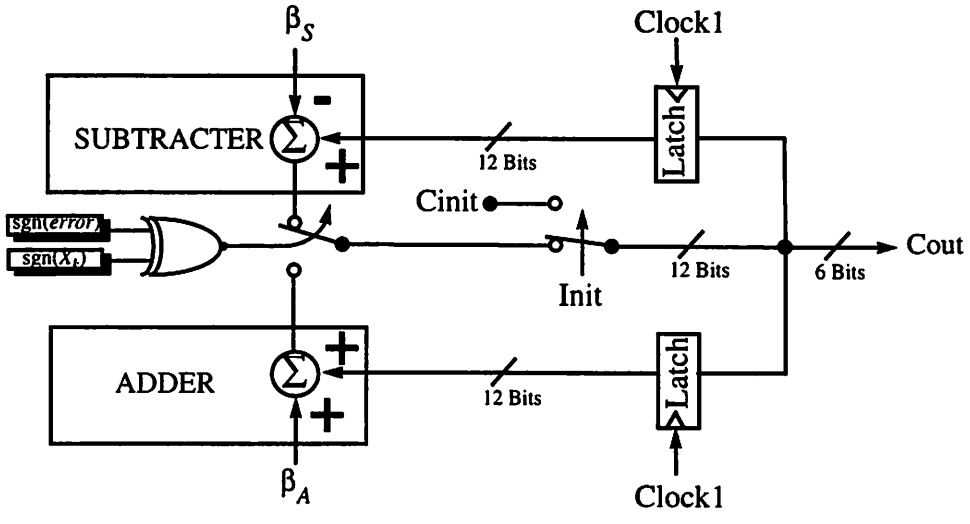


Figure 4.6 Coefficient Update Block.

The arrangement of the above update block as it appears in the layout is shown in figure xx.xx. Twelve bits of the coefficients are stored and used for the next update. However, only six bits are feed to the input of the filter. Beta is represent with six bits which are feed in from off chip, another

six bits for beta is hard wired to ground to correspond to the addition with the 12 bits coefficient (see

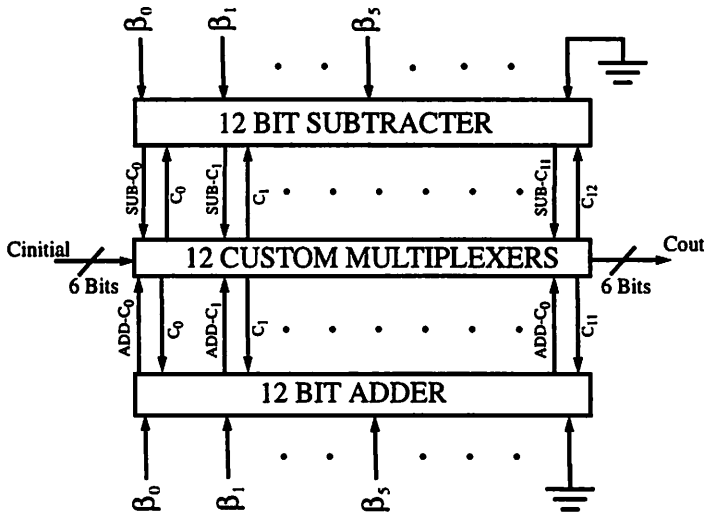
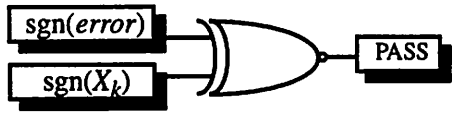


Figure 3.7 Layout of the coefficient update block.

4.3.1 NXOR Gate Design

To implement the function of sgn(*error*).sgn(*Xk*) a custom NXOR gate was design using static CMOS logic. The output of the gate is used in each update coefficient block as a control bit which selects one of two possible coefficients Ck + β or Ck - β. A logic diagram with a truth table is shown in Figure 4.8.



sgn(*error*)	sgn(X_k)	PASS	Function
0 (-)	0 (-)	1	$C_k + \beta$
0 (-)	1 (+)	0	$C_k - \beta$
1 (+)	0 (-)	0	$C_k - \beta$
1 (+)	1 (+)	1	$C_k + \beta$

Figure 4.8 Function of the NXOR gate.

Transistor sizes for the NXOR gate were selected so as to maximize the speed through the coefficient update path. We can see that the loading at the output node and intermediate nodes can be reduced if drain source areas are minimized in M? and M?. After clock1 goes high the error sig-

nal is generated form the decoder outlined in section 4.x. However, the sign of the sample is made immediately available to the input of the NXOR gate, this fact was exploited in the gate design. Therefore, the sign of the sample (which is nothing more that then MSB of the X_k) was feed to an input where the devices were significantly scaled down compared to the sign of the error input. This was done to reduce the amount of drain capacitance in the NXOR gate, thus allowing the output to switch more rapidly. Also, the sign of the sample X_k input was placed closer to the gate output with the large devices placed close to the supply for lower resistance while discharging the output.

4.3.2 Design of the switch network

A three to one multiplexer was design to route the correct set of coefficients to the output. This was done using a network of transmission gates (see Figure 4.9). Essentially the circuit can be thought of as working in two different phases, start-up and normal operation. When the signal labeled as INIT is held high new coefficients are feed into the update block. However, under normal operating conditions INIT is held low allowing a selection, based on the value of PASS,

between $C_k^i + \beta^i$ and $C_k^i - \beta^i$, where C_k^i is the ith bit of the kth coefficient and β^i is the ith bit of beta added to the corresponding coefficient bit.

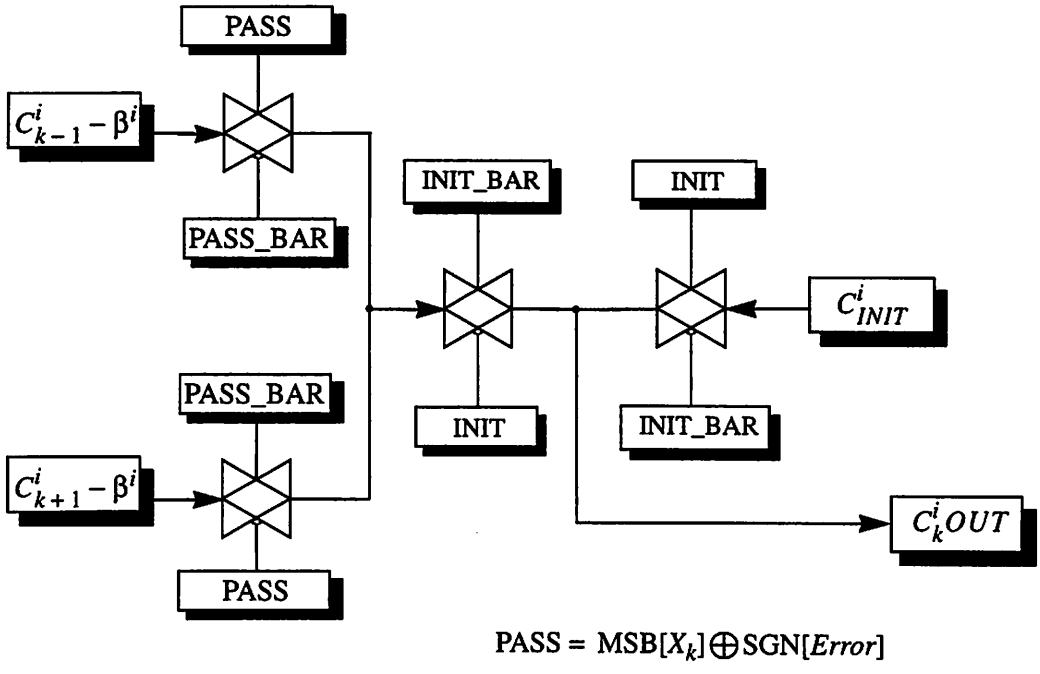


Figure 4.9 Bit level logic for the coefficient multiplexer.

A circuit schematic is given in Figure 4.10. Here we see that the INIT signal disables the buffer for the coefficients coming form the added and subtractor. Thus, only new initialized coefficients are passed to the node labeled as $C_k^i OUT$, the same node that feeds the input latches for filter1. Now under normal operation INIT is low and the buffer consisting of M_9, M_{10}, M_{11}, and M_{12} is enabled leaving the selection of the next coefficient to the control of the PASS signal.

Transistors sizes were again selected so as to minimize the delay in the feedback path of the new coefficients. Therefore, M1 through M4 along with the transmission gates associated with the update coefficients were made minimum size. All the transistors used to initialize the coefficients were scaled either up or down to reduce the loading on the feedback path of the update coefficients. Accordingly, M_9 and M_{12} have high aspect ratios to reduce the series resistance in the output inverter, Both the PMOS and NMOS devices affiliated with the INIT transmission gate are minimum size to again reduce the contributing drain capacitance at the output node.

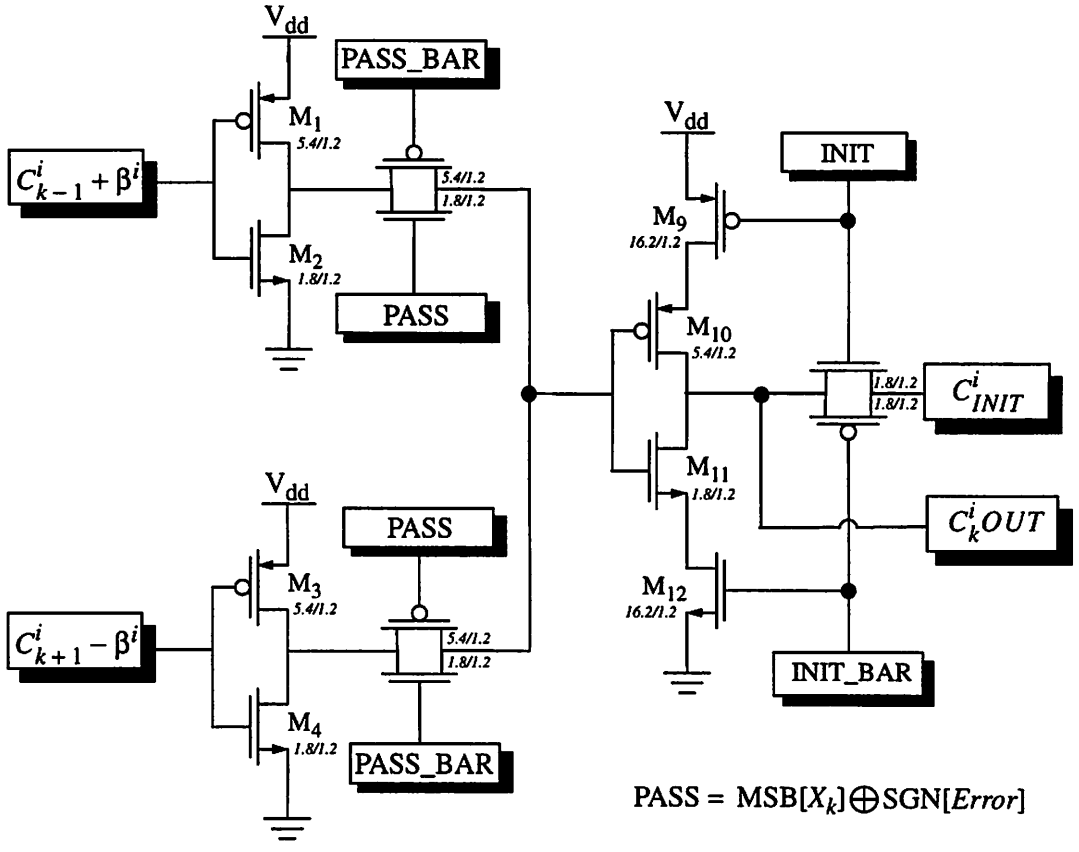


Figure 4.10 Circuit Schematic for the Multiplexing Network.

4.3.3 Coefficient Scaling

To increase the resolution of the coefficients stored each period 11bits were stored in each of the coefficient update blocks while only 6bits are passed to the input of filter1 (see figure 4.x) Additional increases in resolution were obtained by scaling the lower value taps to be on the order of higher value coefficients. For example the middle coefficient for class four partial response is typically the largest, for this filter C4 it is approximately 2.4 when scaled to a unity pulse, however C3 and C5 are -0.8 with the rest of the coefficients on the order of 0.004. C4 is the largest of all the coefficients allowing us to scale up the remaining coefficients which will have the net effect of increasing the resolution of the lower order taps. To implement this scaling a simple left shift of all

the bits is equivalent to multiply the coefficient by two. For this design Figure 4.11 illustrates how each coefficient was scaled to maximize the resolution of all lower order coefficients.

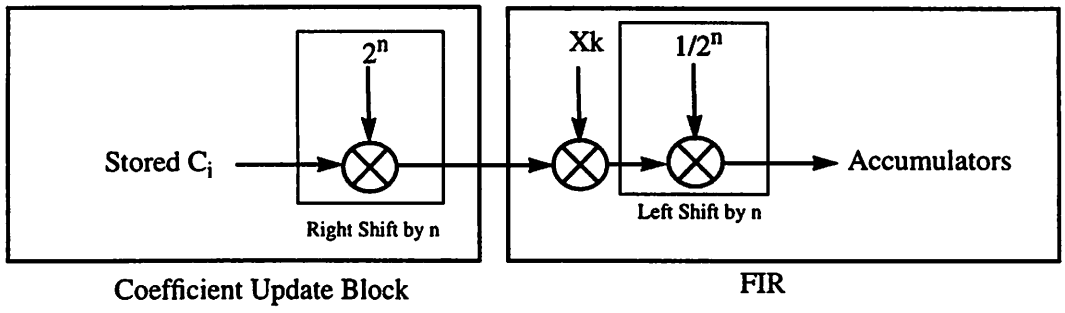


Figure 4.11 Coefficient scaling in the coefficient update blocks.

Scaling of the lower order coefficients was performed by hardwiring the required amount of left shifts in the initial coefficients that were applied. Also, the Beta input to each block was left shifted by the same scaling factor of the corresponding coefficient. The scaled six bit coefficients are latched into filter1. An equal number of right shifts are performed after the multiplication in the FIR filters to again scale the output of all the filters back down by the correct amount before being passed to the accumulators.

4.4 Summary

This chapter was presented to give some of the specifics surrounding the design of the circuits used to update the individual coefficients. The coefficients are updated using only the output from filter1. To reduce the latency in the feedback path the sign LMS algorithm was used. Because only the sign of the error is needed a simple fast static CMOS decoder was designed to perform the operations of slicing and subtracting at the output of filter1. The error signal generated at the output of the decoder feeds eight coefficient update blocks which span the distance of the chip. Therefore, a custom buffer was designed to minimize the delay introduced by a considerable amount of capacitive loading seen at this node.

Each tap was supplied with a coefficient update block. Included in this block are a custom multiplexer, a carry look a head adder and subtractor, along with a custom XOR gate. In designing

each of these blocks reducing the latency in the feedback path was the main consideration. To increase the resolution of each tap the lower value coefficients were scaled up to ensure all coefficients were approximately the same order of magnitude. Further increase in coefficient resolution was achieved by storing 11bits of each coefficient which was used to update the new coefficients.

One of the inputs to the update block was a two input XOR gate which computed sgn(error)sgn(Xk). A custom multiplexer was also designed to The sgn(error) was inputting to each update with

CHAPTER 5

Testing

5.0 Testing Objective

In designing the test setup for the DSP two objectives were to be fulfilled. First, it was desired to exercise all the DSP functions including feeding two six bit digital samples into the filter, adjusting the initial coefficient values and the ability to set and vary the value of Beta. In addition to testing the DSP, it was also desired to interface the adaptive equalizer with the Uehara analog front end chip allowing testing of the entire read channel. Recall from chapter 1 that the Uehara chip processes data using three parallel channels each running at 33MHZ with an effective output rate of 100MHz. However, the DSP samples data using two parallel channels each running at 50MHz also with an effective 100MHz throughput. Therefore, to interface the two chips a data conversion circuit was designed to take three parallel channels running at 33MHz and convert it to two parallel channels running at 50MHz (Figure 5.1). Other features include the ability two measure the current to different portions of the DSP, thus, vacillating the calculation of power.

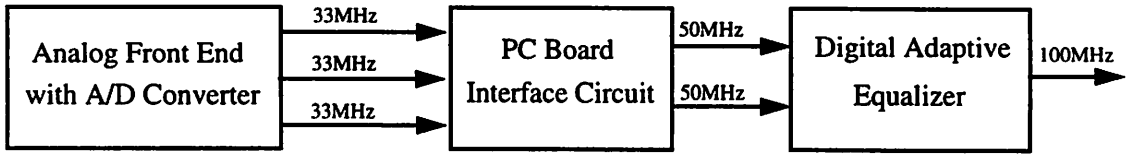


Figure 5.1 Interface Required between Analog front end and the Adaptive Equalizer.

With the first shipment of the Adaptive Equalizer ICs a wire wrap board was built. However, for 100MHz data rates the capacitance associated with wire wrap board prohibited high speed testing. Therefore, the decision was made to create a custom printed circuit board which would accomplish the objective of fully testing the DSP at 100MHz, along with the circuitry necessary to interface the front end chip with the DSP. A two layer double supply plane board was design and laid out at Berkeley after which it was fabricated at Multeck. Description of the board and the test procedure follow.

5.1 Board Design

Shown in Figure 5.2 is a floor plan for the test board. From the figure we can see the desire to keep the DSP test switches and input separate from the interface circuitry needed for the frequency conversion. Separation was desired to facilitate modular testing of the DSP and the read channel. Located in the center on the left hand side is a 128 pin PGA socket to hold the DSP. Space on the right hand side of the board has been used for the latches and multiplexers necessary for the conversion circuitry. Input from the analog front end chip is supplied via a 18 bit bus plug-in shown on the right hand side of Figure 5.2. Four on board supplies are powered up via a supply shown at the bottom. Two clocks are supplied at the top of the board, one for the DSP Master Clock and the other for the interface clocking. While there are many details to the board design only the main components are described in the following sections.

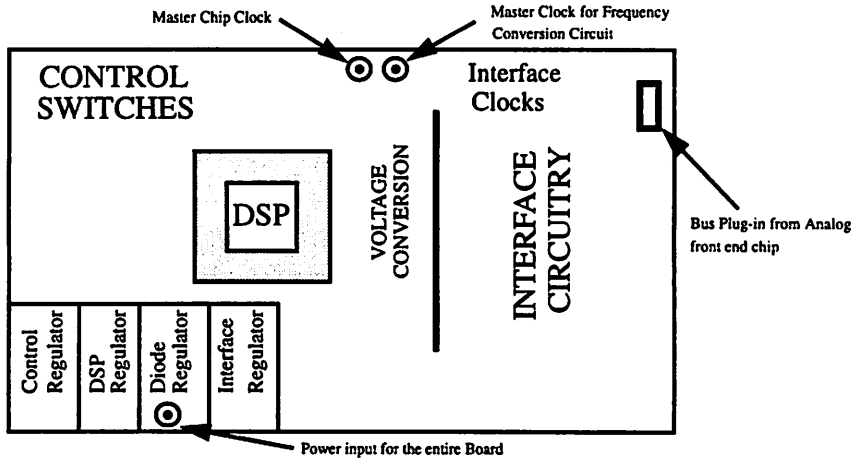


Figure 5.2 DSP test board

5.1.1 Regulator Design

Four adjustable on board regulators were designed to create stable supplies which are required for repeatable test. Two supply planes were provided one of which was used for ground while the other was used to route all four supplies. Add to each section of the board was routed by separating one level of the board into separate supply planes.

To maximize the speed of the conversion circuitry an individual 5 volt supply was used for the standard TTL parts located in this section of the board. Another 3.3v supply was used to feed all the supplies on the DSP exclusively. A third supply was used by all the control logic for the DSP. A fourth and final regulator, generates 2.5volts to bias circuitry for a 5v to 3.3v conversion, this circuit is explained in detail 5.1.2.

From Figure 5.3 we see a circuit diagram of the on board regulator. A LM317T is used to generate a 1.2 volt reference with one resistor placed across this reference along with a potentiometer in series. By varying the 2k pot. we get the desired output voltage. This circuit acts similar to a Vbe multiplier. Ceramic and electrolytic capacitors are used to suppress both high and low fre-

quency noise. The output voltage(Vout) is then channeled to the desired section of the board via the Vdd supply plane.

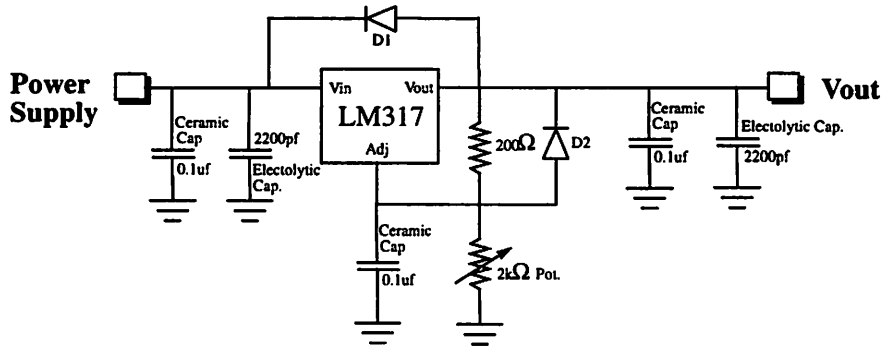


Figure 5.3 Voltage Regulator where Vout is the regulated voltage.

5.1.2 Voltage conversion from (5.0 to 3.3volts)

As explained earlier, the output of the interface circuitry is run at 5 volts to meet the speed requirements of 100MHz. However, the DSP is a 3.3v IC. Therefore, a circuit was needed to provide a conversion between the 5.0 volt section of the board and the 3.3volt DSP. A simple resistor divider was considered to shift the logic levels. At 100MHz low values of R would have been needed to reduce the effect of any parasitic capacitance. Its was estimate that the R would have had to have been on the order of 200ohms which would require considerable current from the output drivers of the TTL gates. Therefore, a similar scheme was developed for the level shifting the 5 volt signal using a simple resistor and a diode. From Figure 5.4 we can see that the output of the interface circuitry will be level shifter from 5v to (Vdiode + 0.7v) or approximately 3.3volts if Vdiode = 2.6v.

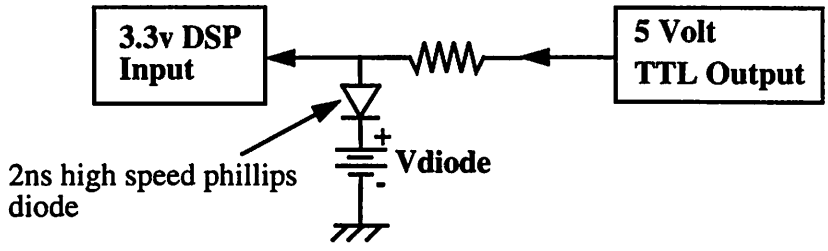


Figure 5.4 Simple circuit to perform voltage level shifting between interface circuit and the DSP.

5.1.3 33MHz to 50MHz conversion

The Analog front end chip makes use of three parallel channels each running at 33MHz. However, the DSP receives two parallel samples at 50MHz. Therefore, the following frequency conversion circuit was included on the test board. Here data from the analog front end is latched into a series of TTL gates. Six samples from the analog front end are stored at any given time. The output of the latches are then feed to six multiplexers which are clock at 50MHz, thus selecting the next sample to be inputted to the DSP. The data is then passed through the voltage conversion circuit before being latched on to the DSP. Figure 5.5 is a block of the frequency conversion block.

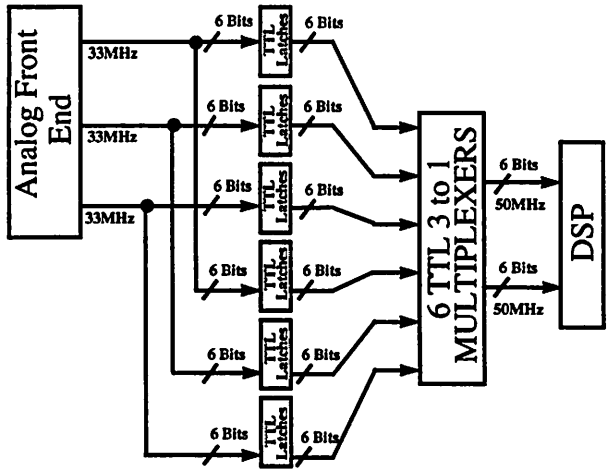


Figure 5.5 Frequency conversion circuit.

5.2 Test Procedure

To facilitate testing an HP16500 Logic analyzer was used to exercise all the functions on the DSP. Data for the Logic analyzer was obtained from Ptolemy simulations. Input sample Vectors were loaded into the logic analyzer. Both the clock and the samples were generated from the analyzer and inputted to the DSP board. The initial coefficients and Beta were adjusted via a bank of switches found on board.

Data was inputted from the Logic Analyzer after all on board setting were fixed. For the first sixteen periods the initialization signal was held high to allow the loading of all initial coefficient states. After initial vectors were loaded, test vectors were then applied to the DSP filter input while the outputs of all four filters were monitored along with the sign of the error and the update coefficient vectors.

A set of 200 vectors were run through the filter. If it was found that after the first 200 samples the filter coefficients were converging on a value the Logic analyzer was then placed in the repeat mode and the same 200 samples were cycled through to study the stability of the filter coefficients over time. The chip was defined as being fully functional when *the coefficients were able to converge and remain stable within 200 sampling periods.*

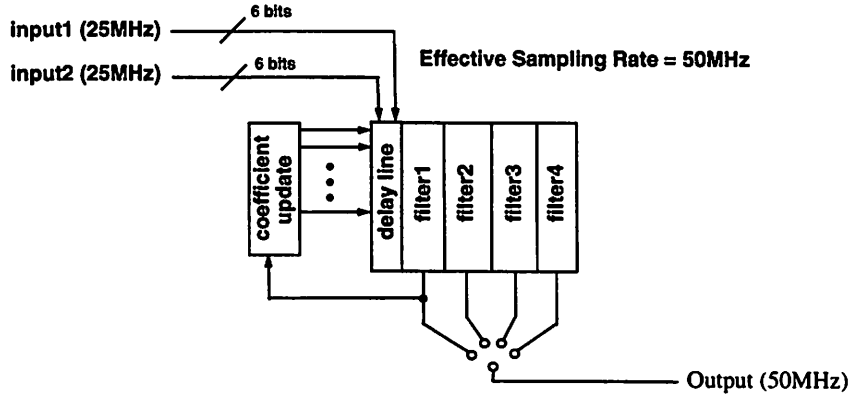
The test procedure was repeated for different values of initial coefficients. Also, the frequency and supply voltage were varied to study the effect on convergence of the filter. After functionality of the filter was shown for a particular test case the Logic analyzer was run in the repeat mode and current measurements were made and used to calculate the power consumption of the DSP. Data and results for all test conditions are shown in the next chapter of this thesis.

CHAPTER 6

Results and Conclusions

6.0 Results of functionality test

Initial test on the DSP were run to verify functionality. Recall from chapter 5.0 that the DSP was defined as being functional if the filter outputs emitted the correct value and the coefficients converged to their target value while remaining stable. Also, recall that the input signals are applied via two parallel channels. Thus, to avoid confusion I will refer to the effective applied frequency of the input signal. For example, an effective 50MHz signal is actually two 25MHz parallel data samples being applied to the DSP.



Our testing began with the simplest case of running a series of DC signals through the filters to confirm the operation of the multipliers and accumulators in the FIR structure. At DC, the filters and the error generation circuit were fully operational. Next, the Logic Analyzer was attached to the DSP and test vectors were applied to the input of the filters at a 10MHz effective sampling rate. Again, the functionality of the filter and the stability of the loop were both verified. The frequency of the input signal was then increased to 20MHz, 40MHz, and 50MHz effectively. At these frequencies the chip was confirmed to be completely functional for both a 5 and 3.3v supply. Finally the DSP was increased to a 100MHz effective sampling rate. With a 5v supply the DSP was confirmed to be both fully functional with the coefficients remaining stable. However, at 100MHz when the DSP supply voltage was decreased to 3.3v the filter coefficients immediately became unstable. *Thus, at 100MHz with a 3.3v supply the DSP was shown to be nonfunctional.* An explanation of the chip failure under these conditions will be outlined in section 6.3.

6.1 Power data

After the functionality of the DSP was confirmed, current measurements were made on all of the chip supplies. These included the supply to filters 1& 2, filters 3 & 4, the supply to the Viterbi sequence detector, a supply to all clocks, and finally the supply to the coefficient update section of the IC. The Logic analyzer was then set in the repeat mode to allow a constant input of samples.

Power data was obtained as a function of supply voltage and frequency. In figure 6.1 and 6.2 we see the results of the power data measurements.

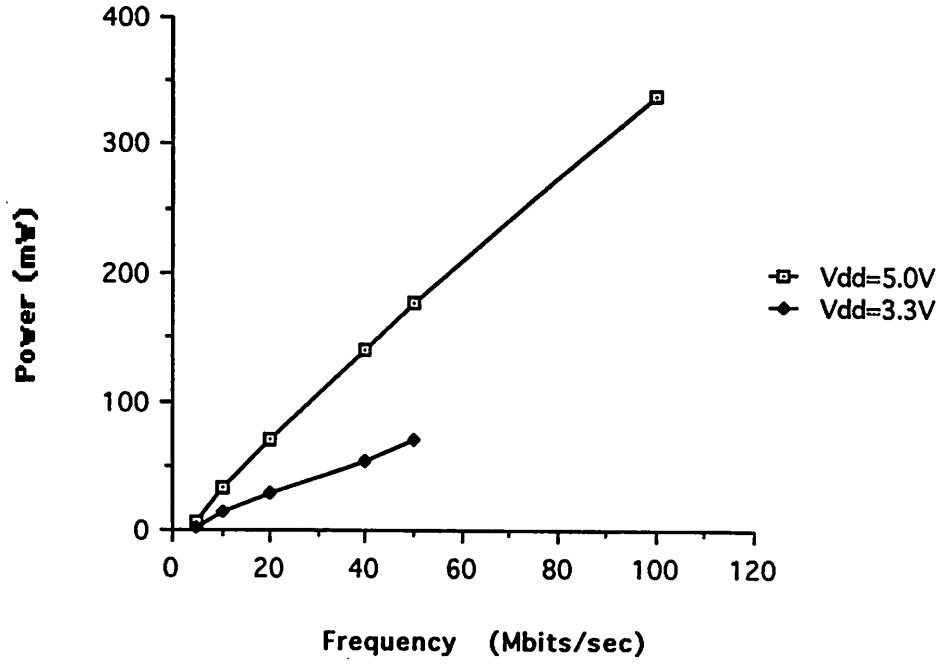


Figure 6.1 DSP power as a function of the effective Frequency.

Here we can see the well know relationship between power and frequency. Notice the linear dependence of the power as a function of supply voltage. From figure 6.1 we can see that the power obeys the well know relationship.

$$P_{ave} = CV^2f \tag{6.1}$$

Further data was obtained to see the relationship between the supply voltage and power dissipation, results are shown in figure 6.2. Here, we can see the dependence of the power on the voltage squared.

A comprehensive list of all the data taken which includes a complete breakdown of the power consumed by each circuit block as a function of frequency and voltage are given in Appendix B.

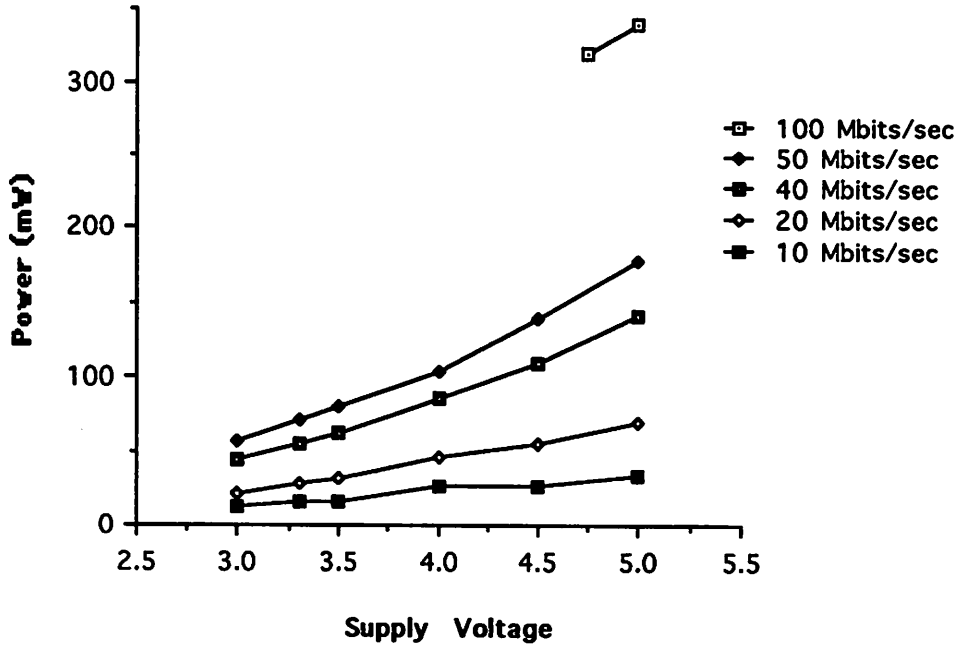


Figure 6.2 DSP Power as a function of the supply voltage

6.2 Description of DSP failure at 3.3v / 100MHz

The adaptive equalizer was designed to be functional for 100MHz at 3.3v. As already mentioned, the DSP was functional at 100MHz with a 5v supply. However, at 3.3v, the filter coefficients became immediately unstable. This was attributed to an unexpected delay in the feedback path of the error generation signal. Recall from chapter 4.0 that the most critical circuit design for the entire adaptive equalizer rested in properly feeding back the sign of the error in the required 40ns period of CLOCK1. From Figure 1.1 we see that the input samples and the coefficients are passed through eight multipliers after which they are latched at the end of this period and passed to the accumulators. Now the output of the multipliers have one period of clock1 (40ns) to pass through the accumulators and into the decoder which produces the sign of the error. During this same clock period the sign of the error must then feed eight XOR gates which will select $C_k + \beta$ or $C_k - \beta$, the results of this selection are then passed to the input latches of the filter1 where the new coefficients must be stable on the falling edge of clock1. It was determined after a series of experiments that the new coefficients were not ready to be latched at the filter input in 40ns. Therefore, an unexpected delay occurred somewhere along the path from the input of accumulators around

the feedback network back to the input of the filter1. The exact location of the error was determined by a series of experiments.

Each circuit block on the DSP has a separate supply, this feature was exploited to isolate the problem associated with the extra latency in the update network. The chip was run at 5.0v, 100MHz to verify functionality, then the supplies to each block on the chip were individually lower until a failure occurred. First, with all blocks at 5.0v, the supplies to the on chip clocks were lowered. The clocks and the equalizer remained functional until the clock supply reached 3.8volts. The failure of the chip with the clock supply at 3.8v and the rest of the chip at 5.0v was attributed to the forward biasing of a junction between one of the 5.0v blocks on the chip and the clock section. However, because forward biasing a junction is normal the clocks were completely ruled out as the problem associated with update signal. Next, the supply to the coefficient update section was lowered with the other supplies remaining at 5.0v. Again, the coefficient update network remained stable until the supply to this portion of the chip fell below 3.8v. The failure here was also attributed to forward biasing of a junction from a 5.0v block to the coefficient update blocks. All supplies were returned to 5.0v and the chip was again supplied with a data pattern until the coefficients became stable about their target value. Finally, the supply to the filters was gradually lowered, however, unlike the previous trials the chip experienced a failure at approximately 4.6volts. Thus, the filters were now the primary suspect of the latency problem.

To further confirm the latency problem was associated with the filters the supply to the this portion of the chip was set just above the failure point, at 4.7v. Then, the supply to the coefficient update section was gradually lowered from 5.0v. When the update supply was lowered to around 4.5v the coefficients again became unstable. This suggests that a considerable delay is introduced by the filters when the supply to this section is lowered to 4.7volts. When just a little more delay is added from lowering the supply in the coefficient update section, which is also along the critical path, the chip immediately fails. However, when the supply to the filters is held at 5.0volts the

update section of the chip can be lowered to 3.8v. Thus, it was believed that the extra delay in the feedback of the error generation signal was associated with the accumulation blocks.

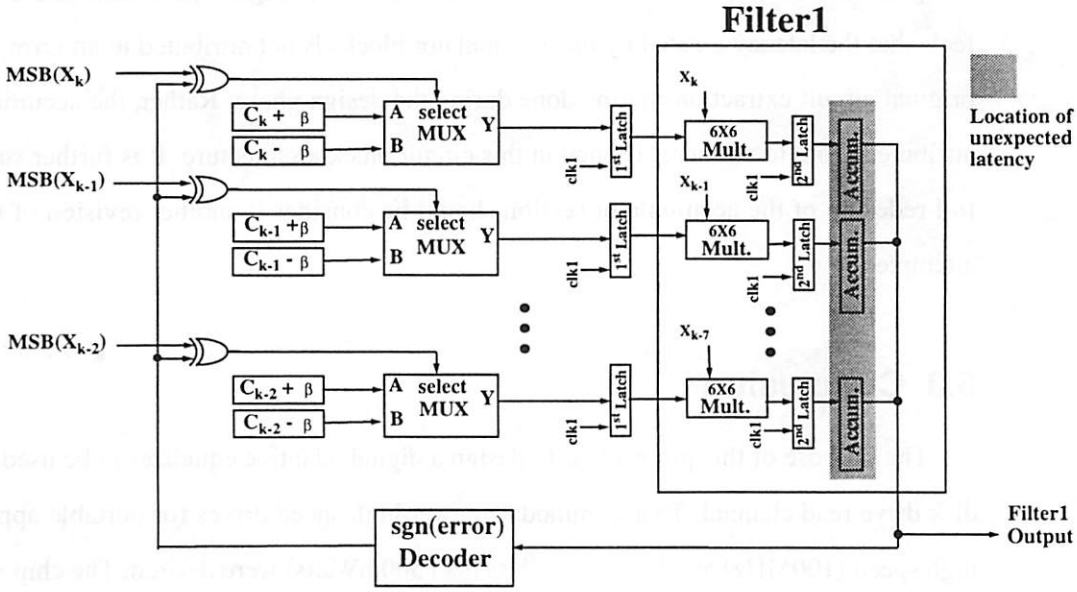


Figure 6.1 Unexpected latency in the accumulators leading to instability in updating the filter coefficients.

The final goal was to isolate the problem inside the filter. It was believed that the latency problem was associated exclusively with the accumulators in the filter section. To verify this assumption, the supply to the filters was lowered to 3.3v with the coefficients frozen in one position. Then at 100MHz, the output of the filters were verified to produce the correct output value for the given coefficients and input samples. This confirmed that the latency problem *was not attributed* to the multipliers found in the filters because this would suggest that a failure occurred before the multiplier output was latched at the accumulator input. Thus, the multipliers were eliminated from consideration. Also, because data at 3.3v was being processed correctly to the output of the filters it was believed that the problem was associated with an *unexpected delay in the accumulators and not a failure in this circuit block.* It will also be noted that a previously published paper [1] suggests that this extra delay was caused by an error in the capacitance estimation done during the layout circuit extraction which lead to an underestimation of the capacitance at the accumulator output thus leading to a conservative simulation of the delay in this area. However, the author of this thesis feels that if the capacitance were improperly extracted this would have created a more

noticeable latency problem in other portions of this chip. In particular, an underestimation of the capacitance at the output of the error generation decoder would have created a catastrophic delay along this critical path which was shown not be the case through experimentation. Thus, the author feels that the latency created by the accumulator blocks is not attributed to an error created by the original circuit extraction routine done during the design phase. Rather, the accumulator delay is attributed more to an inconsistency in this circuit block architecture. It is further suggested that a full redesign of the accumulator section should be consider if another revision of this chip were attempted.

6.3 Conclusion

The purpose of this project was to design a digital adaptive equalizer to be used in a magnetic disk drive read channel. To accommodate newer high speed drives for portable applications both high speed (100MHz) and low power (less than 500mWatts) were desired. The chip was fabricated with a 1.2um CMOS process. All design and testing was done at Berkeley. The major themes, results, and conclusions for this research project are outlined below.

- Both pipelining and parallelism can be used to achieve high data rates for channel equalization. In particular, four parallel filter stages were used to effectively increase the data throughput by a factor of four. For our application, four 25MHz filters were used to obtain a 100MHz throughput.
- To increase the stability of the feedback signal used to update the coefficients the sign LMS algorithm was used rather than the straight LMS algorithm. Although the sign LMS algorithm results in a more course estimate of the new coefficients a considerable savings in hardware is achieved because less computation is required for each update. Fewer operations to update the coefficients implies less latency in the feedback of the error signal and greater overall stability to the filter.
- The DSP was demonstrated to be functional at low frequency (up to 50MHz) with a 3.3volt supply. However, due to an error in the accumulator blocks the filter was not functional at 3.3v, 100MHz data rate. It is believed that this filter could work properly with another design revision because functionality was shown at 5.0volts with 100MHz data throughput.

References

[1] G.T. Uehara, C.S.H. Wong, J.C. Rudell, and P.R. Gray, "A 50MHz 70mW 8-Tap Adaptive Equalizer/Viterbi Sequence Detector in 1.2μm CMOS", Custom Integrated Circuits Conference, San Diego, CA, 1994

[2] H. Kobayashi and D.T. Tang, "Application of Partial-response Channel Coding to Magnetic Recording Systems," *IBM Journel of Research and Development*, July 1970, pp. 368-375.

[3] J.M. Cioffi et. al., " Adaptive Equalization in Magnetic-Disk Storage Channels," *IEEE Communications Magazine*, Feb. 1990, pp. 14-28.

[4] W.L. Abbot, J.M. Coiffi, and H.K. Thapar, "Channel Equalization Methods for Magnetic Storage," *Proceedings of 1989 IEEE International Conference on Communications*, Boston, MA, June 1989.

[5] H.K. Thapar, P. A. Ziperovich, and R.W. Wood, "On the performance of symbol and fractionally-spaced Equalization in Digital Magnetic Recording," *Eigth Conference on Video, Audio and Data Recording*, Birmingham, April 1990

[6] F. Dolivo, "Signal Processing for High-Density Digital Magnetic Recording," Published in the *Proceedings of COMBEURO*, Hamburg, W. Germany, May 8-12, 1989.

[7] W.L. Abbott, J.M. Cioffi, and H.K. Thapar, "Performance of Digital Magnetic Recording with Equalization and Offtrack Interference, *Proceedings of 1989 IEEE International Conference on Communications*, Boston, MA, June 1989.

[8] A. J. Armstrong and J.K. Wolf, "Coded Partial Response Signaling with Peak Detection," San Diego, CA, Dec. 2-5, 1990

[9] J.J Moon, and L. R. Carley, "Partial Response Signaling in a Magnetic Recording Channel," *IEEE Transactions on Magnetic*, Vol. 24, Nov. 1988, pp. 2973-2972.

[10] K. Fisher, J. Cioffi, and H. Thapar, "Modeling Thin-Film Storage Channels," *InterMag.* 1989.

[11] J.D. Coker, et. al. "Implementation of PRML in a Rigid Disk Drive", *IBM Journel of Research and Development*', July 1991

[12] G.R. Grimmett and D.R. Strirzaker, "Probability and Random Processes", pp. 70, Oxford University Press, New York 1982.

[13] E.A. Lee and D.G. Messerschmitt, "Digital Communications", chapter 9-10, Academic Publishers, Boston 1988.

[14] P. M. Clarkson, " Optimal and Adaptive Signal Processing", CRC Press, Boca Raton 1993 test

[15] C.S.H Wong, Masters Thesis, Filed with the Electronic Research Laboratory, UC Berkeley, Summer 1993.

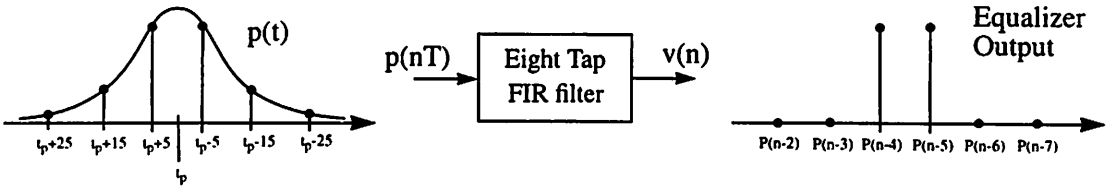
[16] G.T. Uehara, Ph.D. Thesis, Filed with the Electronic Research Laboratory, UC Berkeley, Summer 1993.

[17] J.M. Rabaey, "Design and Analysis of Digital Circuits", UC Berkeley, Fall 1992.

Appendix A

Convergence of the adaptive filter depends on a reasonable guess for the initial set of coefficients. Therefore, calculations were made to estimate the initial values for our coefficients to use for both in Ptolemy simulations and as initial test vectors in the lab. The procedure for calculating these coefficients values will be outlined below.

Recall that the main purpose of the adaptive filter is to reduce the effects of intersymbol interference. Assuming the channel is linear we can find the initial coefficients by analyzing an individual lorentz pulse. From figure A.1 we see that the pulse p(t) will contribute components to symbols both before and after the current symbol period. Now, for class IV partial response we want each transition on the disk to be interpretide as two consecutive symbols of the same sign and magnetude thus creating a (1+D) effect. Thus, for each pulse we want the FIR filter to remove the intersymbol interference and output only two consecutive symbols with a phase delay as shown in figure A.1. The goal then of the adaptive equalizer is to remove all of the ISI.



Again assuming the channel is linear we can use the equation for the lorentz pulse to find the exact amount of ISI at each point in time. For the sample calculation shown it will be assumed that *p(t)* has been normalized to give a 1 at $p(t_s+5)$ and $p(t_s-5)$ where t_s occurs at the peak energy of the lorentz pulse(or $t=0=t_s$), also a PW_{50}/T of 2.25 will be used throughout these calculations. Thus, we have the modified equation for the lorentz pulse,

$$p(t) = \frac{1.198}{1 + (2\frac{t}{22.5})^2}$$

Where T=10 is the sampling period with channel samples being taken at t=nT. The desired samples of *p(t)* are at t +/- 5 in figure A.1. The interfering effect to each adjacent symbol was then calculate and is tabulated in table A.1.

Table A.1

Cont. Time	Discrete Time (n)	p(t)
t_s+75	n+7	0.0263
t_s+65	n+6	0.0348
t_s+55	n+5	0.0480
t_s+45	n+4	0.0704
t_s+35	n+3	0.1121
t_s+25	n+2	0.2016
t_s+15	n+1	0.4309
t_s+5	n	1.0
t_s-5	n-1	1.0
t_s-15	n-2	0.4309
t_s-25	n-3	0.2016
t_s-35	n-4	0.1121
t_s-45	n-5	0.0704
t_s-55	n-6	0.0480
t_s-65	n-7	0.0348

The FIR filter coefficients can be calculated using information about the input pulse and the desired output waveform. We know that the filter output is simply the convolution of *p(t)* with the impulse response of the FIR filter *h(n)*. This can be expressed in matrix form as

$$\begin{bmatrix} p(ts+5) & p(ts-5) & \cdots & p(ts-75) \\ P(ts+15) & p(ts+5) & & \\ \vdots & & \ddots & \\ & & & \\ p(ts+75) & & & p(ts+5) \end{bmatrix} \cdot \begin{bmatrix} C1 \\ C2 \\ C3 \\ C4 \\ C5 \\ C6 \\ C7 \\ C8 \end{bmatrix} = \begin{bmatrix} v(n-1) \\ v(n-2) \\ v(n-3) \\ v(n-4) \\ v(n-5) \\ v(n-6) \\ v(n-7) \\ v(n-8) \end{bmatrix}$$

Now the matrix can be completed with information from table A.1 and figure A.1 and solved for the filter coefficients.

$$\begin{bmatrix} 1 & 1 & 0.4309 & 0.2017 & 0.1121 & 0.0704 & 0.04804 & 0.03483 \\ 0.4309 & 1 & 1 & 0.4309 & 0.2017 & 0.1121 & 0.0704 & 0.04804 \\ 0.2017 & 0.4309 & 1 & 1 & 0.4309 & 0.2017 & 0.1121 & 0.0704 \\ 0.1121 & 0.2017 & 0.4309 & 1 & 1 & 0.4309 & 0.2017 & 0.1121 \\ 0.0704 & 0.1121 & 0.2017 & 0.4309 & 1 & 1 & 0.4309 & 0.2017 \\ 0.04804 & 0.0704 & 0.1121 & 0.2017 & 0.4309 & 1 & 1 & 0.4309 \\ 0.02635 & 0.04804 & 0.0704 & 0.1121 & 0.2017 & 0.4309 & 1 & 1 \\ 0.02062 & 0.02635 & 0.4804 & 0.0704 & 0.1121 & 0.2017 & 0.4309 & 1 \end{bmatrix} \cdot \begin{bmatrix} C1 \\ C2 \\ C3 \\ C4 \\ C5 \\ C6 \\ C7 \\ C8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This is now solved for the initial coefficient values corresponding to a channel with a PW_{50}/T=2.25

$C1$=-0.02448 $C3$=-0.0001462 $C5$=2.04390 $C7$=-0001449
$C2$=-0.006641 $C4$=-0.7249500 $C6$=-724960 $C8$=-0.006742

Appendix B

Decimal #	Binary Number A6 A5 A4 A3 A2 A1	Symbol	Error
31	0 1 1 1 1 1	1	1
30	0 1 1 1 1 0	1	1
29	0 1 1 1 0 1	1	1
28	0 1 1 1 0 0	1	1
27	0 1 1 0 1 1	1	1
26	0 1 1 0 1 0	1	1
25	0 1 1 0 0 1	1	1
24	0 1 1 0 0 0	1	1
23	0 1 0 1 1 1	1	0
22	0 1 0 1 1 0	1	0
21	0 1 0 1 0 1	1	0
20	0 1 0 1 0 0	1	0
19	0 1 0 0 1 1	1	0
18	0 1 0 0 1 0	1	0
17	0 1 0 0 0 1	1	0
16	0 1 0 0 0 0	1	0
15	0 0 1 1 1 1	1	0
14	0 0 1 1 1 0	1	0
+1 13	0 0 1 1 0 1	1	0
12	0 0 1 1 0 0	1	0
11	0 0 1 0 1 1	0	1
10	0 0 1 0 1 0	0	1
9	0 0 1 0 0 1	0	1
8	0 0 1 0 0 0	0	1
7	0 0 0 1 1 1	0	1
6	0 0 0 1 1 0	0	1
5	0 0 0 1 0 1	0	1
4	0 0 0 1 0 0	0	1
3	0 0 0 0 1 1	0	1
2	0 0 0 0 1 0	0	1
1	0 0 0 0 0 1	0	1
C 0	0 0 0 0 0 0	0	1
-1	1 1 1 1 1 1	0	0
-2	1 1 1 1 1 0	0	0
-3	1 1 1 1 0 1	0	0
-4	1 1 1 1 0 0	0	0
-5	1 1 1 0 1 1	0	0
-6	1 1 1 0 1 0	0	0
-7	1 1 1 0 0 1	0	0
-8	1 1 1 0 0 0	0	0
-9	1 1 0 1 1 1	0	0
-10	1 1 0 1 1 0	0	0
-11	1 1 0 1 0 1	0	0
-12	1 1 0 1 0 0	0	0
-13	1 1 0 0 1 1	-1	1
↓ -1 -14	1 1 0 0 1 0	-1	1
-15	1 1 0 0 0 1	-1	1
-16	1 1 0 0 0 0	-1	1
-17	1 0 1 1 1 1	-1	1
-18	1 0 1 1 1 0	-1	1
-19	1 0 1 1 0 1	-1	1
-20	1 0 1 1 0 0	-1	1
-21	1 0 1 0 1 1	-1	1
-22	1 0 1 0 1 0	-1	1
-23	1 0 1 0 0 1	-1	1
-24	1 0 1 0 0 0	-1	1
-25	1 0 0 1 1 1	-1	0
-26	1 0 0 1 1 0	-1	0
-27	1 0 0 1 0 1	-1	0
-28	1 0 0 1 0 0	-1	0
-29	1 0 0 0 1 1	-1	0
-30	1 0 0 0 1 0	-1	0
-31	1 0 0 0 0 1	-1	0
-32	1 0 0 0 0 0	-1	0

Appendix C

	100MHz	
Vdd	5V (mW)	4.75V
Filters 1&2	90.75	88
Filters 3&4	99.25	93.2
Viterbi	36.6	34.5
Clocks	68.5	63.65
Update blocks	43.65	40.6
TOTAL	338.75	319.95

All numbers indicate measured power in mWatts.

	50MHz					
Vdd	5V	4.5V	4.0V	3.5V	3.3V	3.0V
Filters 1&2	50	40.36	26.6	22.61	19.5	16.68
Filters 3&4	54	39.78	31.28	23.41	20.7	16.77
Viterbi	18.1	14.26	11.04	8.26	7.32	5.91
Clocks	34.3	27.09	21.2	15.78	14	11.25
Update blocks	21.5	16.83	13.08	9.76	8.81	7.02
TOTAL	178.15	138.38	103.2	79.73	70.45	57.63

	40MHz					
Vdd	5V	4.5V	4.0V	3.5V	3.3V	3.0V
Filters 1&2	38.6	29.5	25.32	16.66	13.101	12.3
Filters 3&4	41	32.08	24.68	18.585	16.63	13.29
Viterbi	14.75	11.56	8.92	6.65	5.97	4.77
Clocks	27.6	21.73	16.6	12.67	11.55	9.09
Update blocks	17.55	13.5	10.4	7.91	7	5.64
TOTAL	139.55	108.3	86.04	62.47	54.25	45.09

	20MHz					
Vdd	5V	4.5V	4.0V	3.5V	3.3V	3.0V
Filters 1&2	19.35	14.94	16.48	8.57	8.44	5.58
Filters 3&4	20.25	16.24	12.56	9.24	8.83	6.72
Viterbi	7.3	5.8	4.48	3.32	3	2.4
Clocks	13.7	10.89	8.28	6.3	5.676	2.82
Update blocks	8.65	6.84	5.24	3.88	3.49	4.65
TOTAL	69.25	54.72	47.04	31.32	29.007	22.17

	10MHz					
Vdd	5V	4.5V	4.0V	3.5V	3.3V	3.0V
Filters 1&2	8.95	7.47	10.44	4.79	5.08	4.41
Filters 3&4	10.15	8.1	6.32	4.55	4.15	3.36
Viterbi	3.65	2.88	2.24	1.645	1.48	1.2
Clocks	6.9	5.44	4.16	3.15	2.8	2.25
Update blocks	4.3	3.37	2.64	1.96	1.74	1.41
TOTAL	33.95	27.27	25.8	16.1	15.27	12.63

Appendix D

The following figure is the actually Ptolemy facet used to simulate all of the non-idealites for the proposed adaptive equalizer architecture. The facet used to generate the PRIV channel samples is not shown.