New and Improved Detection Methods for Digital Magnetic Recording

by

Thomas Anthony Conway

Submitted for the degree of
Doctor of Philosophy

to the
University of Limerick

Supervisor: Dr. John Nelson

Submitted to the University of Limerick June 1996
Declaration

I hereby declare that this thesis is entirely my own work and has not been submitted as an exercise for a degree, in part or full, to any other University.

Thomas A. Conway

August 23, 1996
Acknowledgments

I would like to especially acknowledge the guidance and assistance of my supervisor Dr. John Nelson. My sincere thanks is due for supervising this work through from the original proposal with tremendous enthusiasm, interest and insight.

I also wish to thank Analog Devices B.V. for providing funding for this project. In particular, thanks is due to Peter Real and Philip Quinlan for their help and interest in the work.

Others who have helped include:
Dave McCormack for helping get the early versions of LINUX working and pointing out the value of \TeX\ for documentation,
Stephen Bergin for help with the IC design tools,
Dr. Sean McGrath for the use of test equipment,
Dr. Oliver McCarthy for his useful and interesting DSP modules,
Richard Conway for his interest and discussions,
Eamonn McGuinness for help with PC/Network support, and all the members of the Telecommunications Research Center.

I wish to acknowledge the help of the Eurochip Services for IC fabrication and design tool support.

I also wish to acknowledge the work of the LINUX authors and GNU free software foundation in making excellent software available.

And finally, thanks to my family and friends for their support throughout.
New and Improved Detection Methods for Digital Magnetic Recording

Thomas A. Conway

Abstract

The magnetic recording channel can be viewed as a communications channel in which the signal read back is corrupted by noise and impairments in the channel. Practical detection of the recorded data with the smallest probability of error is of primary concern. To this end, partial response and sampled data channels with Viterbi detection have recently become prominent in the magnetic recording industry.

In this thesis, new and modified detection systems are presented that perform better on the recording channel than existing detectors. The improvements are made through better coding structures and novel detector architectures that allow practical implementation at the required data rates. Performance gains are verified through theoretical analysis and computer simulations.

A rate 6/8 matched spectral null code for a PR4 channel has been developed. A VLSI implementation of this detector is designed, fabricated and prototype measurements presented.

Matched Spectral Null (MSN) coding for the PR1 response on the magnetic recording channel is proposed and demonstrated to achieve significant gains.

A novel coding scheme for PR4 based channels is proposed which provides a moderate coding gain but with an identical coding rate to existing channels. This provides increased performance at moderate recording densities.

A new channel target and matching detector is proposed for use on high density recording channels. This detector outperforms existing and proposed detectors while requiring modest complexity.

The performance of the new detectors are compared with existing detection systems such as PR4, EPR4 and DFE and other proposed advanced detectors such as FDTS/DF. The results show that the proposed schemes compare favorably in terms of error rate performance and implementation complexity.
# Table of Contents

0.1 Symbols ................................................. 1  
0.2 Abbreviations ........................................ 2  
0.3 Preface to Thesis .................................... 3  

1 Digital Magnetic Recording .......................... 5  
  1.1 Introduction ......................................... 5  
  1.2 Applications of Digital Magnetic Recording ...... 5  
  1.3 Channel Characteristics ......................... 6  
     1.3.1 Channel Impairments ......................... 7  
  1.4 Optimum Detection .................................. 8  
     1.4.1 The Viterbi Algorithm ....................... 9  
     1.4.2 Application of MLSD ......................... 11  
  1.5 Practical Detection ................................ 11  
     1.5.1 Peak Detection ................................. 12  
     1.5.2 Linear Equalization ......................... 14  
     1.5.3 Partial Response Channels .................. 14  
     1.5.4 DFE Systems ................................. 16  
  1.6 Magnetic Recording System Architecture ...... 17  
     1.6.1 Write Section ................................. 17  
     1.6.2 Read Section ................................... 19  
     1.6.3 Continuous Time Filter ....................... 20  
     1.6.4 LMS Equalizer ................................. 21  
     1.6.5 Gain and Timing Loops ....................... 22  
     1.6.6 Detectors ..................................... 23  
  1.7 Conclusions .......................................... 26  

2 Channel Model and Analysis ......................... 28  
  2.1 Introduction ......................................... 28  
  2.2 System Overview .................................... 28  
  2.3 Channel Model ....................................... 29  

II
2.4 Power Spectral Density of Data Source .............................................. 30
2.5 Equalization ................................................................................. 32
   2.5.1 Zero Forcing Equalizer ......................................................... 32
   2.5.2 Minimum Mean Square Error Equalizer ................................. 33
2.6 Choice of Desired Response .......................................................... 35
   2.6.1 Partial Response Systems ...................................................... 35
   2.6.2 Decision Feedback Equalizer ............................................... 36
2.7 Detector Performance ................................................................. 37
   2.7.1 Threshold Detection .............................................................. 37
   2.7.2 Detection for Partial Response Systems ................................. 38
   2.7.3 Performance of Viterbi Detector with Noise Correlation ............ 40
   2.7.4 Performance of EPR4 Viterbi Detector ................................. 42
   2.7.5 Performance of PR4 Viterbi Detector ..................................... 45
   2.7.6 Decision Feedback Detection ............................................... 47
   2.7.7 FDTS/DF Detector ............................................................... 49
2.8 Matched Filter with MLSD ............................................................ 51
2.9 Computer Model ......................................................................... 54
2.10 Calculations and Simulations ....................................................... 55
   2.10.1 Electronics Noise ................................................................. 55
   2.10.2 Electronics Noise and Media Noise ..................................... 56
2.11 Comparison of Detectors .............................................................. 57
2.12 Conclusions ................................................................................. 59

3 Matched Spectral Codes ...................................................................... 61
   3.1 Introduction ................................................................................. 61
   3.2 Matched Spectral Null Principle .................................................. 61
      3.2.1 Application to Magnetic Recording ..................................... 61
   3.3 DC Free Codes ........................................................................... 62
      3.3.1 Finite Sequences ................................................................. 62
      3.3.2 Infinite Sequences .............................................................. 64
      3.3.3 Sample DC Free Code ......................................................... 65
   3.4 Time Varying Trellis Structure ..................................................... 67
      3.4.1 Run Length Constraints ....................................................... 68
   3.5 Higher Order Nulls ..................................................................... 68
      3.5.1 Example Higher Order DC Null Code ................................... 70
   3.6 Partial Response Channels ........................................................... 71
      3.6.1 The Dicode Channel ........................................................... 72
3.6.2 Combined Trellis ........................................... 72
3.6.3 Catastrophic Sequences ..................................... 73
3.7 Time Varying Trellis ........................................... 74
  3.7.1 Accumulated Distance Between Unmerged Paths .............. 74
  3.7.2 Non-Catastrophic Codes ................................... 77
  3.7.3 Other Time Varying Structures ................................ 77
3.8 Performance of MSN Codes on the Recording Channel .......... 78
3.9 Matched Spectral Codes on the PR1 channel ..................... 79
  3.9.1 Probability of Error for PR1 Detector ......................... 81
  3.9.2 Model Results .............................................. 82
  3.9.3 Simulated Channel Model .................................... 83
  3.9.4 Simulation Results .......................................... 83
  3.9.5 More Efficient Codes ....................................... 84
  3.9.6 Application of MSN Coding on the PR1 Channel to Tape Drives .... 85
3.10 Conclusions ..................................................... 85

4 Design and Implementation of MSN VLSI IC .......................... 87
  4.1 Introduction ...................................................... 87
  4.2 Rate 6/8 Code .................................................... 87
  4.3 Decoder Trellis ................................................... 88
  4.4 Practical Aspects of the Detector ................................. 90
    4.4.1 ADC Resolution Requirements ............................... 90
    4.4.2 Finite Precision Arithmetic .................................. 91
    4.4.3 Path Memory ................................................. 93
    4.4.4 Synchronization .............................................. 93
  4.5 VLSI Prototype .................................................. 94
    4.5.1 Transmit Section ............................................. 94
    4.5.2 Viterbi Detector and Receive Section ....................... 95
    4.5.3 Viterbi Detector ............................................. 96
    4.5.4 Path Memory ............................................... 98
    4.5.5 Receive Section ............................................. 100
    4.5.6 Support and Test Logic ..................................... 101
    4.5.7 Design Methodology ....................................... 103
    4.5.8 Design for Test ............................................. 103
    4.5.9 Prototype IC Specifications ................................ 103
  4.6 Prototype Measurements ......................................... 103
    4.6.1 Functional Test ............................................ 104
5 Distance Lengthening Codes

5.1 Introduction ................................................................. 110
5.2 Losses Due to Noise Correlation ................................. 110
5.3 Minimum Distance Errors on the Decode Channel ......... 112
5.4 Parity Codes to Reduce Noise Correlation ................. 113
  5.4.1 Performance of Parity Codes .................................... 114
  5.4.2 Run Length Properties of Parity Codes ................ 115
  5.4.3 Complexity of Implementation ............................. 116
5.5 Modulo RDS Codes ..................................................... 116
  5.5.1 Rates of Modulo RDS Codes ................................ 118
  5.5.2 Encoding and Run Length Properties of Modulo RDS Codes . 118
  5.5.3 Example Rate 16/18 $|RDS|_3 = 0$ Code .................. 120
  5.5.4 RDS Modulo 3 Codes Path Memory Requirements .......... 120
  5.5.5 Performance of Modulo RDS Codes .................... 123
  5.5.6 Complexity of Implementation .......................... 123
5.6 Error Events .............................................................. 124
  5.6.1 Error Events with Modulo 3 RDS Codes .......... 125
5.7 Comparison with MLSD detector ................................. 126
5.8 Detailed Rate 16/18 Encoder/Decoder ....................... 126
  5.8.1 Encoder .............................................................. 127
  5.8.2 Algorithmic Mapping ................................ ........ 129
  5.8.3 Channel Polarity and Codeword Inversion ............ 129
  5.8.4 Run length constraints ................................ 130
  5.8.5 Correction and Decoding .................................. 131
  5.8.6 Codeword Correction ................................ .... 131
  5.8.7 Codeword Decoding ......................................... 133
  5.8.8 Algorithmic Decoding ................................ .... 133
  5.8.9 Logic Implementation Complexity ..................... 134
  5.8.10 Viterbi Path Memory ....................................... 134
5.9 Conclusion .............................................................. 135
6 High Density Detector

6.1 Introduction .................................................. 137
6.2 1+D Channel with Decision Feedback ................... 137
6.3 Theory of Operation ........................................ 137
   6.3.1 Feedback Decisions .................................. 139
   6.3.2 Feedback filter ....................................... 140
   6.3.3 Target Response ...................................... 140
6.4 Detector Update Equations ................................. 141
   6.4.1 Path Memory Updates ................................ 143
   6.4.2 Feedback Update ..................................... 145
6.5 Practical Consideration for Detector .................... 145
   6.5.1 Update Equations .................................... 147
   6.5.2 Detector Pipelining .................................. 150
6.6 Control Loops ................................................ 152
6.7 Line Coding ................................................... 155
   6.7.1 Sample Code Constraint ............................. 156
   6.7.2 Existing Line Codes .................................. 157
6.8 Theoretical Performance .................................... 158
   6.8.1 Sensitivity to α Parameter .......................... 160
   6.8.2 Comparison with PR4 and EPR4 Detectors ........... 162
6.9 Simulated Performance of Detector ...................... 163
6.10 Comparison with Other Detectors ....................... 164
6.11 Conclusions ................................................ 165

7 Summary and Conclusions ..................................... 166

7.1 Introduction .................................................. 166
7.2 Context of Thesis .......................................... 166
7.3 Contributions of the Thesis .............................. 167
   7.3.1 MSN Codes ........................................... 167
   7.3.2 DL Codes ............................................. 167
   7.3.3 New High Density Target Response OPDDC .......... 168
7.4 Potential Applications of Proposed Schemes ............. 168
7.5 Comparison of Existing and Proposed Detectors ........ 169
7.6 Future Work .................................................. 171
7.7 Conclusions .................................................. 173

References ......................................................... 174

VI
A Authors Publications

B Mathematical Derivations
   B.1 Frequency Response of the Lorentz Channel ......................... B 1
   B.2 Probability of Error for EPR4 Detector ........................... B 2

C Channel Model Simulation Software
   C.1 Software Description ................................................. C 1
      C.1.1 Platforms ...................................................... C 2
      C.1.2 Performance ................................................... C 3
   C.2 Software Functionality ................................................ C 3
      C.2.1 Lorentz Channel and Basic Parameters ........................ C 3
      C.2.2 Data Source ................................................... C 3
      C.2.3 Continuous time filter ........................................ C 3
      C.2.4 Target Responses ............................................... C 4
      C.2.5 Detectors ....................................................... C 4

D Bit Error Rate Confidence Interval ........................................ D 1

E MSN IC
   E.1 Representative Schematics ........................................... E 1

F Rate 16/18 RDS mod 3 code
   F.1 VERILOG Description of Encoder ................................... F 1
   F.2 VERILOG Description of Corrector ................................ F 5
   F.3 VERILOG Description of Decoder ................................... F 7
   F.4 Synthesized Logic ................................................... F 9
## 0.1 Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impulse at $t = 0$</td>
<td>$\delta(t)$</td>
</tr>
<tr>
<td>Unit step at $t = 0$</td>
<td>$u(t)$</td>
</tr>
<tr>
<td>Recording density</td>
<td>$\rho$</td>
</tr>
<tr>
<td>Minimum distance</td>
<td>$d_{\text{min}}$</td>
</tr>
<tr>
<td>Branch metric for the value $x$ at time $k$</td>
<td>$\lambda_x^k$</td>
</tr>
<tr>
<td>Path metric for the state $x$ at time $k$</td>
<td>$\Gamma_x^k$</td>
</tr>
<tr>
<td>Unit time delay</td>
<td>$D$ or $z^{-1}$</td>
</tr>
<tr>
<td>50% Width of Lorentz pulse</td>
<td>$PW_{50}$</td>
</tr>
<tr>
<td>Recorded bit period</td>
<td>$T_{\text{bit}}$</td>
</tr>
<tr>
<td>Expected value of random variable $x$</td>
<td>$E[x]$</td>
</tr>
<tr>
<td>Fourier transform of $x(t)$</td>
<td>$X(f)$</td>
</tr>
<tr>
<td>Discrete time Fourier transform of $x_k$</td>
<td>$X(e^{j2\pi f T})$</td>
</tr>
<tr>
<td>$D$ transform of $x_k$</td>
<td>$X(D)$</td>
</tr>
<tr>
<td>Cumulative Gaussian function $\frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\alpha^2/2} d\alpha$</td>
<td>$Q(x)$</td>
</tr>
<tr>
<td>Effective SNR for bit error rate $BER$</td>
<td>$20 \log_{10} Q^{-1}(BER)$</td>
</tr>
</tbody>
</table>
### 0.2 Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Compare Select</td>
<td>ACS</td>
</tr>
<tr>
<td>Alternating Running Digital Sum</td>
<td>ARDS</td>
</tr>
<tr>
<td>Analog to Digital Converter</td>
<td>ADC</td>
</tr>
<tr>
<td>Automatic Gain Control</td>
<td>AGC</td>
</tr>
<tr>
<td>Bit Error Rate</td>
<td>BER</td>
</tr>
<tr>
<td>Decision Feedback Equalizer</td>
<td>DFE</td>
</tr>
<tr>
<td>Distance Lengthening</td>
<td>DL</td>
</tr>
<tr>
<td>Error Control Coding</td>
<td>ECC</td>
</tr>
<tr>
<td>Finite Delay Tree Search</td>
<td>FDTS</td>
</tr>
<tr>
<td>Finite Impulse Response</td>
<td>FIR</td>
</tr>
<tr>
<td>Infinite Impulse Response</td>
<td>IIR</td>
</tr>
<tr>
<td>Intersymbol Interference</td>
<td>ISI</td>
</tr>
<tr>
<td>Least Mean Squares</td>
<td>LMS</td>
</tr>
<tr>
<td>Matched Filter Bound</td>
<td>MFB</td>
</tr>
<tr>
<td>Matched Spectral Null</td>
<td>MSN</td>
</tr>
<tr>
<td>Maximum Likelihood Sequence Detection</td>
<td>MLSD</td>
</tr>
<tr>
<td>Mean Square Error</td>
<td>MSE</td>
</tr>
<tr>
<td>Minimum Mean Squared Error</td>
<td>MMSE</td>
</tr>
<tr>
<td>Most Significant Bit</td>
<td>MSB</td>
</tr>
<tr>
<td>One Plus D with DC feedback</td>
<td>OPDDC</td>
</tr>
<tr>
<td>Partial Response Maximum Likelihood</td>
<td>PRML</td>
</tr>
<tr>
<td>Phase Locked Loop</td>
<td>PLL</td>
</tr>
<tr>
<td>Run Length Limited</td>
<td>RLL</td>
</tr>
<tr>
<td>Running Digital Sum</td>
<td>RDS</td>
</tr>
<tr>
<td>Signal to Noise Ratio</td>
<td>SNR</td>
</tr>
</tbody>
</table>
0.3 Preface to Thesis

The magnetic recording channel can be viewed as a communications channel in which the signal read back is corrupted due to impairments in the channel. Detection of the recorded data with the least probability of error is of primary concern. To this end, partial response channels and sampled data channels have recently become prominent in the magnetic recording industry. While these channels provide better error rate performance and higher recording densities than traditional detection systems, they are considerably more complex. Future channels may provide even better error rate performance and/or higher recording densities.

In this thesis, new and improved detection systems are presented that perform better on the recording channel than existing detectors. The improvements are made through new coding structures and novel detector architectures that allow practical implementation at the required signal rates. Performance gains are verified through theoretical analysis and computer simulations. The suitability for VLSI realization is addressed through prototypes and analysis.

In chapter 1, the basic operation of the magnetic recording channel is outlined and details of existing detection schemes and data channel architectures are presented.

Chapter 2 describes an analytical model of the channel and detection system for a number of existing detection schemes. A software model is also described that allows the channel and detectors to be simulated under various operating conditions. These allow calculation of the performance of the various schemes and bit error simulations to be performed. They also provide a basis for calculating and verifying the performance of the proposed schemes in the later chapters.

Chapter 3 considers the implementation of matched spectral null codes that have been proposed as a possibility for future recording channels. Some of the problems associated with such codes are described and resolved. The performance of such schemes on the magnetic recording channel are assessed and their use with the PR1 partial response is proposed as a method of increasing linear recording density.

Chapter 4 uses the methods developed in chapter 3 to design a rate 6/8 matched spectral null code for the PR4 channel. All the implementation details are considered and a VLSI implementation of this detector is designed, fabricated and prototype measurements presented.

Chapter 5 presents a novel coding scheme for PR4 based channels. The method achieves less coding gain than matched spectral null codes, but achieves a higher coding rate. The code rates achievable are similar to existing line code rates and hence provide an increase in performance over existing channels, but without the requirement to operate at higher
channel rates or recording densities. The principles behind the codes are described and the practical details of a particular rate 16/18 code are presented.

Chapter 6 proposes a new target response based on the PR1 channel and decision feedback. A suitable detector is developed that is readily implementable. The new target is particularly suitable for high density recording channels and achieves better performance than other proposed detectors while being less computationally intensive. Such a detector and target will be useful for future channels where high recording densities are required.

The thesis is concluded in chapter 7, and future work is identified for the presented schemes and related areas.
Chapter 1

Digital Magnetic Recording

1.1 Introduction

Magnetic recording channels store information by orienting the magnetic particles of suitable magnetic media in particular directions. The orientation is maintained by the media over time and can be read back at a future time to recover the stored information. The write process is carried out with an inductive head that generates a magnetic field when a current is passed through it. Almost all digital magnetic data storage uses saturation recording where the writing magnetic field saturates the magnetic medium in one of two possible directions.

The read process may use an inductive head (may be the same as the write head) or a magneto-resistive head. The inductive head responds to the time derivative of the magnetic flux due to the medium. Magneto-resistive heads are normally shielded to respond to the perpendicular component of the magnetic flux and so deliver a voltage that corresponds to transitions in the recorded magnetic field. However, the voltage does not depend on the velocity of the medium as it does with an inductive head.

For the purpose of analysis, the recording channel may be viewed as a communications channel, with a signal transmitted into the channel at one time and received out of the channel at some later time, with various impairments. Storage densities are essentially determined by distances between transitions and head geometries. However, it is usual to assume that the magnetic medium is moving with a constant velocity and translate distances into time periods and consider frequencies for convenience.

1.2 Applications of Digital Magnetic Recording

Magnetic recording is used for the majority of computer data storage, both for online and archival storage. It is also used for video and audio storage, and manipulation particularly in studio environments. The increase in the amount of digital data being stored
and processed has in part been driven by exponential recording density increases. The increases in recording densities have come from continuously increasing performance in all areas of the recording process. This includes higher performance media, smaller and better head geometries including magnetoresistive read heads, lower flying heights and potentially contact recording, narrower tracks with improved servo tracking, better coding efficiency and signal processing and error correction codes. As each of these factors improve the performance of magnetic recording by a small factor, the product of all the improvements results in large increases in recording densities [CAT+90][Sma95].

1.3 Channel Characteristics

In digital magnetic recording, the magnetic medium is magnetized with the write head in either of two directions with the magnetization field sufficiently strong to saturate the medium. When a reversal of the magnetizing field occurs, a transition in the recorded magnetization is written.

While reading, the head responds to the transitions where a change in the recorded field occurs. Because the write transition takes place over a finite distance of the medium, and due to practical limitations on the design of read heads and the spacing between the medium and head, the signal from the read head is a broad pulse.

The magnetic transition is often considered to be in the form of an arctangent to yield closed form mathematical expressions. This assumption in conjunction with a read head model yields a head output voltage, as a function of distance $x$, as a Lorentz pulse [Mal93]

$$E(x) \propto \frac{1}{1 + (2x/PW_{50})^2}$$

(1.1)

where the distance $PW_{50}$ depends on system geometries.

Assuming a constant relative velocity between medium and head, the response of the read head to a single magnetic transition is commonly described as

$$V(t) = \frac{1}{1 + (2t/PW_{50})^2}$$

(1.2)

where $PW_{50}$ is the length of time corresponding to the 50% pulse width. This can be considered as the response to a step in the write current of the record head.

The output wave form for a given read signal can be determined for a given input bipolar pulse train by linear superposition. This is only true if the input is a two level bipolar signal of sufficient level to saturate the magnetic medium. When discussing the density of recording channels, it is common to refer to the density $\rho$ as the ratio of $PW_{50}$ to the recorded bit time $T_{bit}$:

$$\rho = \frac{PW_{50}}{T_{bit}}$$

(1.3)
Fig 1.1 shows the Lorentz pulse response for some values of $PW_{50}$. Typical recording densities of current sample data PRML channels are of the order of 2 bits per $PW_{50}$. For a given medium and head combination, the pulse width $PW_{50}$ is a constant and the recording density is determined by the write pulse width $T_{bit}$.

![Figure 1.1: Lorentz step response](image)

1.3.1 Channel Impairments

In any channel, various forms of noise and interference limit the data rate (or storage density) and error performance. Typical impairments relevant to magnetic channels are as follows.

**Intersymbol interference:** Intersymbol interference is distortion in the read signal due to the finite bandwidth of the channel response. This causes the read pulses to expand in time and interfere with each other. This interference increases rapidly at high densities. In principle, the intersymbol interference can be compensated for if the channel response is known. However, practical limitations require compromises to be made[Jor88].

**Electronics Noise:** Electronics noise generally refers to any noise that comes from the read electronics. This should be dominated by the pre-amplifier input noise and head noise and is normally wide band white noise[Jor88].

**Media Noise:** Media noise is due to the particulate nature of magnetic media. This comes through the read head and is colored Gaussian noise[Jor88].
**Off-track noise:** Off-track noise is interference from adjacent tracks that the read head overlaps due to the finite servo precision. This interference will have a similar spectrum to the read data [Jor88].

**Overwrite noise:** Overwrite noise is noise due to data that was written on the medium previously and has been overwritten. This noise is concentrated at low frequencies [Jor88].

**Non-linear writing:** At high densities the write process becomes non-linear and this causes problems for most detection methods which assume that the channel is linear [NW90]. This can be combated by using non-linear write pre-compensation or using coding to avoid adjacent transitions in the write data.

**Transition Noise:** When a magnetic transition is written, the region where the transition occurs has some uncertainty associated with it. This type of effect leads to additional noise around the transition location and is hence dependent on the written signal [ZW95].

### 1.4 Optimum Detection

From a telecommunication point of view, the optimum detection method for a recording channel can be determined. If a linear channel with Gaussian white noise is assumed, then the optimum detector can be shown to be a sampled matched filter with a noise whitener and maximum likelihood sequence detector [For72]. Fig 1.2 shows a diagram of the detector.

![Diagram of optimum detector for linear channel](image)

**Figure 1.2:** Optimum detector for linear channel

The matched filter [TS86] is a filter that is matched to the impulse response of the channel in the presence of white noise. If a single bit was transmitted, then passing the received signal through the matched filter, taking a single sample and using a threshold detector would be the optimum detector in terms of probability of error. The sampled signal is termed a 'sufficient statistic' because no additional information from the signal can improve the probability of error. However, when the received signal contains ISI, the sampled output of the matched filter contains ISI. The sampling process is carried out at the baud rate and because the filter is a linear filter, the sampled signal is a sufficient
statistic for the data sequence. This sampled signal contains all the information about the
read signal, irrespective of aliasing, if the sample timing is correct. The noise that is in the
signal is not white because it has come through the matched filter. If the channel response
is \( h(t) \), then the matched filter response is \( h(-t) \), the channel response reversed in time.

This sampled signal can now be passed through a noise whitening filter. The output of
this filter is now a baud rate sampled signal that is a sufficient statistic and contains white
noise. Maximum likelihood sequence detection may then be performed.

Maximum likelihood sequence detection (MLSD) involves making a decision about the
received data based on all the samples of the received signal. Given a particular observed
sequence \( \{y_k\} \), the MLSD seeks to choose a channel input sequence \( \{\hat{x}_k\} \) that maximizes
the conditional probability \( P_{Y\mid X}(\{y_k\}\mid\{\hat{x}_k\}) \).

This can be used to detect data in the presence of intersymbol interference. For example,
if 10 binary data bits were transmitted through a channel, it would in principle be possible
to calculate the channel outputs for each of the \( 2^{10} \) possible transmitted sequences. If each
of these outputs were compared to the actual received samples, then the 'closest' one could
be chosen as the most likely transmitted sequence. This brute force method of looking at
\( 2^N \) sequences becomes impractical for any reasonable length data sequence. However, an
algorithm proposed by A. J. Viterbi for decoding convolutional codes has been shown to
allow the practical implementation of MLSD in the presence of ISI [For72].

1.4.1 The Viterbi Algorithm

The Viterbi algorithm is normally cast in the form of a finite state machine. The output
of the state machine is the channel output symbol. This output and the next state are a
combination of the current state of the machine and the current input to the machine. For
the ISI case, the channel output \( y_k \) can be written as

\[
y_k = h_0 x_k + h_1 x_{k-1} + \ldots + h_v x_{k-v} \tag{1.4}
\]

Here, the \( x_k \) are the inputs to the channel and come from a finite alphabet of \( m \) possible
input symbols. The coefficients \( h_k \) model the channel ISI as a FIR filter. The input symbol
is \( x_k \) and the state is \( \{x_{k-1} \ldots x_{k-v} \} \) with \( v \) symbols in the state. Thus, the number
of possible states is \( m^v \). For the magnetic recording channel case, the input is a binary symbol
and hence \( m = 2 \).

The state machine can be redrawn in the form of a trellis diagram as shown in fig 1.3.
Such a diagram indicates all the possible states at time \( k \) and how they related to all the
possible states at time \( k + 1, k + 2, \ldots \). Each branch from any state \( S_a \) at time \( k \) to a state
\( S_b \) at time \( k + 1 \) corresponds to being in state \( S_a \) at time \( k \), receiving a particular input
symbol, outputting a particular channel symbol and ending up in state \( S_b \). If at time \( k \) a
channel symbol $y_k'$ is received then it can be compared to all the possible values of channel output for each state. Some metric can be then attributed to the difference between the expected value $y_k$ and the received value $y_k'$. This metric is known as the branch metric and for a hard decision decoder would be the Hamming distance between the received bits and the expected bits. For a soft decision system with Gaussian white noise the appropriate metric is $(y_k - y_k')^2$ [VO79]. The MLSD requires that the path with the minimum total metric be chosen.

The observation made by Viterbi was that at any time $k$, if the total metrics for each path to a given state were calculated, then the one with the minimum total metric could be chosen and the rest discarded. This path is called the 'survivor path'. If this is done for each state possible then one of these must be the path with the minimum overall metric. So, at the time $k$, for each state the survivor path and its total metric must be stored. When the next channel symbol(s) arrives the new total metrics are calculated by adding the branch metrics to the previous state's total metrics. For each state, all the total metrics of branches ending in that state are considered and the minimum is chosen.

The new survivor paths at time $k$ for each state are the survivor paths corresponding to the chosen minimum metric with the relevant symbol added. In principle, the path memories would have to be infinite in length. However, there is usually a high probability that the survivor paths will merge together within a finite time in the past so the path memory may be finite. The Viterbi algorithm achieves maximum likelihood detection with a complexity that is proportional to $m^v$ with $m$ being the symbol alphabet size and $v$ being the number symbols defining the state.

The benefit of using MLSD is gained from taking advantage of the correlation in the desired sequence compared to the uncorrelated noise. In the white Gaussian noise channel, the branch metric is $(y_k - y_k')^2$. An error event occurs when an incorrect path through the
trellis is chosen over the correct path. The probability of such an error event is determined by the Euclidean distance between any two possible sequences \( \{ \tilde{y}_k \} \) and \( \{ \hat{y}_k \} \):

\[
d = \sqrt{\sum_k (\tilde{y}_k - \hat{y}_k)^2}
\]

The probability of error is governed by the minimum value of \( d \) over all possible sets of distinct sequences. This \( d_{\text{min}} \) can be related to the probability of error through [LM94]:

\[
P_e = K Q\left( \frac{d_{\text{min}}}{2 \sigma} \right)
\]

where \( K \) is a small constant [For72], \( \sigma \) is the noise standard deviation and \( Q() \) is the cumulative error function. The value of \( K \) ceases to be dominant at high SNR.

1.4.2 Application of MLSD

In practice, the number of states to implement MLSD with a recording channel and matched filter would be prohibitive. For this reason, the matched filter is replaced by an equalizer that shapes the channel response to some response for which MLSD can be easily done. For magnetic recording channels some partial responses have been identified as suitable targets to equalize to, and for which MLSD may be performed with a manageable number of states. The PR4 and EPR4 responses are such targets [TP87]. The use of such partial responses with MLSD detection is referred to as PRML and is available for high end disk drives [HWC91][CGK+91].

1.5 Practical Detection

While the optimum detection scheme can be formulated, the practical implementation of detection schemes has to be considered. Typical recording channels are required to operated at data rates up to \( 10^8 \) bits per second or greater. This requires implementation of complete hardware real-time detection schemes. In the case of the optimum maximum likelihood detection, the channel response from the matched filter can be at least 10 symbols or more long. This would require a Viterbi detector with \( 2^{10} \) states which is impractical for implementation.

For this reason, a number of practical suboptimum schemes have been considered. Up to recently, peak detection was the most commonly used scheme on disk drive recording channels. Other detection schemes such as linear equalization with threshold detection have been used on tape drive products with a DC free code [FKSO86]. Presently PRML channels are available and are being used on high end drives [Wel94][YPP94][Cho94][Abb94]. In the following sections, the peak detection method, linear equalizer, partial response and decision feedback type detection systems are outlined.
1.5.1 Peak Detection

Peak detection is a relatively straightforward method of detecting recorded data and can be implemented using analogue building blocks. For peak detection channels, the recording channel is assumed to be a differentiator. The data is stored by writing a transition to represent a logic bit 1 and the absence of a transition represents a logic bit 0. The read signal is low pass filtered with a linear phase filter. Some gain can be applied at frequencies near the top of the passband to compensate for the attenuation of the magnetic recording channel at higher densities.

![Waveforms](image)

**Figure 1.4: Peak detection waveforms**

The output of these equalizing filters is a pulse for each transition in the recorded data. The pulse may be either positive or negative. The peak detector can now recover the data by detecting the presence or absence of a pulse. This is illustrated by the waveforms in fig 1.4. The filtered read signal is compared with a positive and negative threshold and a pulse gate signal is set when it exceeds these levels. This signal indicates the presence of a pulse.

The position of the pulse peak is located by differentiating the filtered read signal and observing when it crosses the zero line. This zero crossing represents the peak position. A pulse, and hence a data bit 1 is determined to have occurred when a zero crossing occurs while the pulse gate is active.

Timing for the recovered data is obtained by using a phased lock loop (PLL). The phase error signal for the loop can be obtained by comparing the position of the qualified zero crossings to the PLL VCO clock edges. The implementation of such peak detectors is
readily achieved using analogue design blocks and complete detection systems are available as single chip devices[Goo92].

While peak detection works well at low to medium recording densities it has some drawbacks. It can be seen that the peak detector operates on a bit by bit basis and ignores information about the read signal such as the fact that each adjacent pulse must be of opposite polarity. The peak detector is sensitive to two kinds of error. Fig 1.5(a) shows a pulse that is missed due to noise reducing its amplitude. Fig 1.5(b) shows a pulse that has its peak shifted into the next bit period due to noise. As the recording density increases, the bit period reduces and pulses begin to interfere with each other. This causes both peak shifts and amplitude reduction. Boosting the high frequencies, can reduce the pulse widths but also boosts the noise.

![Diagram](image)

(a) Missed Bit  
(b) Bit Shift

Figure 1.5: Peak Detection Errors

**RLL Codes**

As the recording density increases, the intersymbol interference increases. The effect of this can be reduced through run length limited (RLL) codes. These RLL codes prevent adjacent logic 1s and hence separate the pulses to avoid interference. The most common ones are the rate 2/3 (1,7) code and the rate 1/2 (2,7) code. The rate 2/3 (1,7) code constraints the data to have at least one logic 0 between each logic 1 and a maximum of 7 logic 0s between each logic 1 for timing recovery purposes. The rate 1/2 (2,7) code constraints the data to have at least two logic 0s between each logic 1 and a maximum of 7 logic 0s between each logic 1. From a straight forward point of view, separating each transition by one logic 0 as in the (1,7) code should allow data to be written twice as densely and result in an overall density increase of $2 \times \frac{2}{3}$ or 33%. The (2,7) code should allow data to be written 3 times as densely and allow an increased recording density of $3 \times \frac{1}{2}$ or 50%. In practice, the rate 2/3 (1,7) code is more commonly used. This is because, as the recording density increases,
the bit period is reduced and the received pulse spans more bit periods. This increases the likelihood of bit shift errors. Hence, the rate $2/3$ $(1,7)$ code achieves a good compromise between bit shift errors and missing bit errors.

Peak detection channels were the most common method of recovering data for hard disk drive channels up to recently.

1.5.2 Linear Equalization

Linear equalization is one of the most straightforward ways of dealing with ISI in a communications channel. This involves placing a filter in the receiver with a response that undoes the ISI introduced by the channel response. If the equalizing filter has a response that is the inverse of the channel response, then the output signal will be identical to the input signal.

Nyquist developed criteria that can be used to design equalizing filters. The main Nyquist criterion for an ISI free channel requires that to transmit a signal with a baud rate $\frac{1}{T}$ with no ISI at the sampling points, the response of the system (channel plus equalizer) must be anti-symmetric about the Nyquist frequency $\frac{1}{2T}$ [Pro89].

In principle, the bandwidth of the system can be reduced to $\frac{1}{2T}$ to minimize the noise. However, this requires a brickwall lowpass filter response with high sensitivity to timing jitter. A raised cosine response is commonly used to provide a compromise between noise bandwidth and timing margins [Pro89].

For a channel with a null in the frequency response such as the magnetic recording channel, it is not possible to provide full equalization. However, linear equalization can be used if a modulation code with a null at DC is used. This method is used in some digital magnetic tape storage systems. These systems are typically helical scan systems where the rotating head is transformer coupled to the stationary read circuits and hence DC write currents cannot be used. DC free codes also allow the use of azimuth recording where adjacent tracks are written with a different azimuth angle thus allowing recording with no guard bands [Mal93].

1.5.3 Partial Response Channels

Partial response channels are channels whose response does not seek to avoid ISI, but rather, allows a controlled and known amount of ISI [KP75]. The effect of the ISI is then accounted for in the receiver. This allows the channel response to be adjusted to achieve other desirable properties. For example, signaling can be achieved with the minimum Nyquist bandwidth of $\frac{1}{2T}$ where $T$ is the symbol duration. The channel response can have nulls at DC and the Nyquist frequency to ease equalization requirements at the band edges.
as in the case of modified duobinary partial response systems.

Various classes of partial response systems exist and are normally characterized by their response in terms of delayed symbols. These are normally written as a polynomial in $D$, a delay of one symbol period. Table 1.1 lists some common partial response systems.

<table>
<thead>
<tr>
<th>Response</th>
<th>Name</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 + D$</td>
<td>PR1</td>
<td>Null at $\frac{1}{2\pi}$</td>
</tr>
<tr>
<td>$1 - D$</td>
<td>Dicode</td>
<td>Null at DC</td>
</tr>
<tr>
<td>$1 - D^2$</td>
<td>Modified Duobinary (PR4)</td>
<td>Null at DC and $\frac{1}{2\pi}$</td>
</tr>
<tr>
<td>$(1 - D)(1 + D)^2$</td>
<td>Extended PR4 (EPR4)</td>
<td>Null at DC and $\frac{1}{2\pi}$</td>
</tr>
</tbody>
</table>

Table 1.1: Example Partial Responses

The output of a partial response channel is a multi-level signal and the original data can be recovered by threshold detection and some logic. However, this results in decreased immunity to noise due to the increased number of signal levels. To overcome this drawback, it was recognized that the multiple levels have inherent correlation and by tracking this correlation with a maximum likelihood detector regains most or all of the loss in noise immunity.

**Application to Magnetic Recording**

It was recognized [Kob71], that some partial responses have a response that is similar to the magnetic recording channel and might offer some detection gain over conventional systems. In [TP87] a number of partial responses of the form $(1 - D)(1 + D)^n$ are considered for their suitability to magnetic recording channels. The most common choice is modified duobinary (PR4). The choice of partial response is determined as a tradeoff between matching the shape of the partial response to the magnetic recording channel and the complexity of the detection system. This can be illustrated by considering the PR4 response of $1 - D^2$ and the EPR4 response of $(1 - D)(1 + D)^2$. The frequency responses of each are shown in figure 1.6.

For a channel with white Gaussian noise, the partial response that fits closest to the actual channel will have the best performance. The recording channel response is different at different recording densities and hence a single partial response is not suitable over a large range of densities. The PR4 response is considered suitable for densities up to 2 while the EPR4 response is a better match at higher recording densities. At present PR4 based detection systems are common with EPR4 also being considered[SYM+93][Ana95].
1.5.4 DFE Systems

An alternative detection system is the use of Decision Feedback Equalizers. These systems can help to overcome ISI without excessive noise enhancement.

**Principle of DFE**

The DFE consists of a forward linear filter and feedback filter as shown in fig 1.7 with representative impulse responses at various points. The forward filter shapes the received pulse into a causal pulse that has only ISI after the main sample of the pulse. This 'post cursor' ISI is then canceled by feeding the bit decisions into the feedback filter which calculates what this ISI is.

The improvement in using such a structure comes from the fact that the post cursor ISI is canceled without any enhancement in the noise and hence performs better than linear equalization. The DFE is non-optimum because a decision is made on a single data sample while the 'energy' in the canceled data is not used. The DFE has been proposed as a suitable detection scheme for the magnetic recording channel[FCAP91] and a commercial implementation has become available [Gra95].

**FDTS DFE**

A variation on the DFE has been proposed to improve on the non-optimum detection of the DFE. The Finite Delay Tree Search with Decision Feedback (FDTS/DF) method[MC90] does not cancel all the post cursor ISI. The data decision is then determined by using a fixed length tree search method based on a number of data samples not canceled. If the
input to the decision tree was determined by a two-sample signal (i.e., all but two samples are canceled), then the decision tree determines which of the 4 possible paths \{1,1\}, \{1,0\}, \{0,1\} or \{0,0\} was most likely and makes a decision on what the first symbol was.

This method uses more of the pulse energy in making the decision and hence should achieve better performance than a DFE.

However, the direct implementation of the tree search requires multiplications to be performed within a single bit period. In [Ken91] the FDTS/DF is analyzed using RLL constrained codes and a simplified detection method is presented for codes with a \( d \geq 1 \) constraint. This simplifies the detector but at the expense of the code rate.

### 1.6 Magnetic Recording System Architecture

In this section the basic components of the recording system are outlined for a sampled data read channel. Current implementations provide all of these components integrated as a single IC.

#### 1.6.1 Write Section

The basic block diagram of the write system is shown in fig 1.8. User data is assumed to be random and is encoded to apply desirable constraints to the signal to be recorded. Typically this line coding would constrain the data to guarantee transitions within suitable intervals to ensure that timing information is available while reading the data at a later stage. The output from the line coder is then precoded. The write circuits may also provide some data dependent timing shifts to the write currents. This is known as write precompensation. In peak detection channels the compensation was chosen to reduce the effect of linear intersymbol interference. A different type of compensation is used in sampled
data channels with the purpose of reducing non-linear effects that can occur when writing at high recording densities [NW90].

In helical scan recording where a rotating head is used, a rotary transformer is normally used to couple the rotating section to the read/write electronics. In this case, the encoder will also need to ensure that the write signal is DC free, otherwise, it will not pass through the coupling transformer.

![Figure 1.8: Magnetic recording write section](image)

**Precoding**

Precoding is a simple encoding process that can be applied to partial response channels to allow bit by bit detection of the transmitted data. Consider the PR4 channel $1 - D^2$. Given an output sample of value 0 then it is required to know whether the last non-zero sample was a +2 or a -2 to correctly decode the transmitted data. This implies a memory requirement in the decoder and means that a single channel error can be propagated into many decoded bit errors (an infinite number if runs of 0 are not constrained).

Let the bits to be transmitted be $a_n$. Let the bits $b_n$ be defined recursively as

$$b_n = a_n \oplus b_{n-2}$$

(1.7)

with the $\oplus$ operation representing exclusive-or and the bits from the set \{1,0\}.

If the bits are represented with a logic 1 represented as +1 and a logic 0 represented as −1 then the previous operation is equivalent to

$$b_n = -a_n b_{n-2}$$

(1.8)

If the symbols $b_n$ are passed through a PR4 channel, then the output is

$$y_n = b_n - b_{n-2} = -a_n b_{n-2} - b_{n-2} = -b_{n-2}(a_n + 1)$$

(1.9)

If the source bit $a_n$ was +1 (a logic 1) the output is +2 or −2, while if the bit $a_n$ was −1 (a logic 0), the output is always 0. Hence, through the precoding operation the transmitted
data can be detected on a symbol by symbol basis and a single channel error will result in a single bit error. This precoding also has the advantage that the polarity of the channel can be inverted and the correct data is recovered. This is useful in the magnetic recording channel as the polarity of the channel is usually difficult to guarantee.

A similar system of precoding can be applied to any partial response channel. PR4 channels are normally precoded to avoid sensitivity to the channel polarity and the PR4 Viterbi detector can be designed to take account of this and provide the correct data.

**Line Coding**

In order to guarantee sufficient transitions for timing, gain control loops and equalizer adaption, line codes are used. These are coding schemes that map unconstrained user data to constrained data. Run length limited (RLL) codes limit the runs of consecutive bits in the constrained data. Peak detection channels use \((d, k)\) codes with \(d \geq 1\) as described in section 1.5.1 but with a rate of at most \(\frac{3}{4}\). For sampled data channels, higher rate codes are desirable as the \(d \geq 1\) is not required. For the PR4 channel a rate 8/9 \((0,4/4)\) code is often used[Wo91]. The 0 constraint allows consecutive 1's to occur. In this code the 4/4 constraint implies the maximum run length of 0's in the constrained data is 4 and when the data is deinterleaved into two streams the maximum run length of 0's in each stream is 4 also. When this code is used with the PR4 precoder, the constraints in the deinterleaved stream limits the required path memory of the Viterbi detector as well as providing non-zero channel samples for timing and gain recovery.

For DFE channels, the deinterleaved constraint is not required. However, the rate 8/9 code is still used due to its high rate and ease of implementation.

The use of trellis coding to increase the performance of partial response channels has also been proposed[WU86]. The most promising techniques are matched spectral null codes based on theory developed by Karabed and Siegel in [KS91]. These are considered in detail in chapter 3.

1.6.2 Read Section

Fig 1.9 shows the read section of a sampled data channel. The signal from the read head, which may be an inductive or a magnetoresistive head, is amplified by a preamplifier which is normally mounted close to the read head assembly. This signal is then amplified by a variable gain amplifier (VGA) which is used as part of the automatic gain control (AGC) loop. The output signal of the VGA is fed to a continuous time filter which lowpass filters the signal and may also provide some equalization of the signal. The signal is then digitized by an ADC and equalized in the digital domain. The filter is typically an adaptive
FIR filter. The equalized signal is used by timing recovery and AGC circuits and used by the detector. In PRML systems, a Viterbi detector is used although other possibilities such as decision feedback equalizers exist [FCA+91].

![Diagram of a magnetic recording read section sampled data channel]

**Figure 1.9: Magnetic recording read section sampled data channel**

### 1.6.3 Continuous Time Filter

In the sampled data channel, a continuous time filter is employed before the data is sampled. Sampling cannot be higher than the channel baud rate due to the high sampling frequencies involved. The continuous time filter provides the functions of band limiting the signal and shaping it to some appropriate response. In the case of a disk drive system, movements of the read head cause the channel characteristics and data frequencies to change between the inner tracks and the outer tracks. For this reason and the fact that the channel characteristics may vary between various disk assemblies, the properties of the filters need to be programmable via electronic means.

In older peak detection channels, linear phase filters were commonly used to avoid pulse distortion. Low pass linear phase filters, with some form of boost at higher frequencies to compensate for the magnetic channel were employed. This trend has been followed in the early implementations of PRML sampled read channels [Wel94] [YPP94] [Cho94].

These type of filters have a transfer function in the form

\[ H(s) = \frac{-ks^2 + \omega_c^2}{LP(s, \omega_c)} \]  \hspace{1cm} (1.10)

where \( LP(s, \omega_c) \) is the denominator of a linear phase lowpass filter with a cutoff frequency at \( \omega_c \). The factor \(-ks^2 + \omega_c^2\) in the numerator provides high frequency gain, programmable
by the factor \( k \). In the frequency domain \( s \to j\omega \) and the numerator becomes \( k\omega^2 + \omega_c^2 \) which is real and thus, does not alter the phase response of the filter.

A typical 7th order equi-ripple low-pass filter with boost has a response of

\[
H(s) = \frac{-ks^2 + 1.3179\omega_c^2}{s^2 + 1.6858\omega_c s + 1.3179\omega_c^2} \cdot \frac{2.5915\omega_c^2}{s^2 + 1.5408\omega_c s + 2.5915\omega_c^2} \times \frac{5.3685\omega_c^2}{s^2 + 1.1453\omega_c s + 5.3685\omega_c^2} \cdot \frac{0.861\omega_c}{s + 0.861\omega_c} \tag{1.11}
\]

Fig 1.10 shows the type of response that such filters achieve with the two programmable parameters cutoff frequency \( \omega_c = 2\pi f_c \) and boost \( k \).

![Figure 1.10: Responses of continuous time filter](image)

It can be seen from fig 1.10 that these linear phase filters have quite a slow roll off due to the requirement for linear phase. This may result in aliasing of high frequency noise into the passband during sampling. This suggests that for sampled PRML channels a non-linear phase filter, such as a Butterworth filter, may be preferable [Son94]. The non-linear phase may be compensated by the digital filter. It is likely that more optimum continuous time filters exist but the existing filters were used due to the mature programmable designs that existed for peak detection filters.

1.6.4 LMS Equalizer

The equalizer in the sampled detection channel is normally a Least Mean Square (LMS) adaptive filter. This is used due to its relatively simple implementation. This equalizer is based on a finite impulse response filter with \( N \) taps. The output of such a filter is based on \( N \) coefficients with

\[
y_k = \sum_{i=0}^{N-1} c_i x_{k-i} \tag{1.12}
\]
where \( x_k \) is the input and \( y_k \) is the output. The adaption algorithm operates on the coefficients \( c_i \) to achieve some desired criteria.

For each output sample \( y_k \) a predetermined training sequence or a tentative data decision is used to calculate the expected ideal output \( d_k \). An error signal \( e_k = d_k - y_k \) can then be calculated. The LMS algorithm seeks to minimize the mean of the square error.

The mean squared error \( MSE \) can be expressed as

\[
MSE = E[e_k^2]
\]

(1.13)

Differentiating with respect to \( c_i \) to find the minimum yields

\[
\frac{\delta MSE}{\delta c_i} = E[2e_k x_{k-i}]
\]

(1.14)

\[
= -E[2e_k x_{k-i}]
\]

\[
= 0 \text{ for a min or max}
\]

\[
\frac{\delta^2 MSE}{\delta c_i^2} = E[2x_{k-i}^2] \geq 0 \iff \min
\]

While an exact solution could be found by solving a set of linear equations, the LMS algorithm approximates \( \frac{\delta MSE}{\delta c_i} \) as \(-2e_k x_{k-i}\) instead of \(-E[2e_k x_{k-i}]\) and iteratively moves towards the minimum. This is achieved by updating each coefficient with

\[
c_i^{k+1} = c_i^k - \mu \frac{\delta MSE}{\delta c_i}
\]

(1.16)

where \( \mu \) is a small constant. With the approximation for \( \frac{\delta MSE}{\delta c_i} \) this becomes

\[
c_i^{k+1} = c_i^k + \mu 2e_k x_{k-i}
\]

(1.17)

The factor \( \mu \) is just a scaling factor and the update can be considered a single multiplication for each coefficient in the filter. The complexity of the filter can then be considered to be of the order of \( 2N \) multiplications for an \( N \) tap filter. The structure of such a filter is shown in fig 1.11. Further simplifications such as considerations of the signs of the error and data samples can be employed to further simplify the implementation.

1.6.5 Gain and Timing Loops

The sampled data channel requires sampling with the correct timing phase and suitable scaled data. This is achieved with a timing recovery loop and automatic gain control loop.

For the timing recovery loop on a PR4 channel, a simple gradient update algorithm is used [CDH92]. The phase error is calculated as

\[
\phi_{\text{error}} = -y_k \hat{y}_{k-1} + y_{k-1} \hat{y}_k
\]

(1.18)
with $y_k$ being the equalized samples and $\hat{y}_k$ being tentative decisions based on $y_k$. This timing error calculation may fail to acquire tracking on startup if the phase error is 180 degrees off. Further modifications may be added to overcome possible problems [DSU89].

For the gain recovery loop on a PR4 channel, a simple gradient update algorithm is also used. Typically the loop gain error is calculated as

$$G_{error} = \hat{y}_k(y_k - \hat{y}_k)$$  \hspace{1cm} (1.19)

From this expression for gain error, it can be noted that when the tentative data decisions $\hat{y}_k$ are zero, the gain error is also zero. When the channel starts up, the signals in the system may be too low to cause non-zero decisions. Hence, some form of analogue gain control is also required to raise the signal to a suitable level so that the digital gain control begins to operate.

1.6.6 Detectors

For PRML channels, the most common target is the PR4 response, with the EPR4 response also being considered. The PR4 response is the most straightforward implementation requiring a 2 state Viterbi detector operating at half the channel rate. The EPR4 detector requires 8 states operating at the full channel rate.

**PR4 Detector**

The PR4 detector can be implemented as two interleaved $1-D$ detectors by alternating between odd and even samples. Each detector is only required to run at half the channel data rate. This allows higher speed of operation. The $1-D$ channel can be cast in the form
of a finite state machine as shown in table 1.2. The states are equivalent to the previous channel input.

<table>
<thead>
<tr>
<th>Input Symbol</th>
<th>State</th>
<th>Output</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → −1</td>
<td>−1</td>
<td>0</td>
<td>−1</td>
</tr>
<tr>
<td>0 → −1</td>
<td>+1</td>
<td>−2</td>
<td>−1</td>
</tr>
<tr>
<td>1 → +1</td>
<td>−1</td>
<td>+2</td>
<td>+1</td>
</tr>
<tr>
<td>1 → +1</td>
<td>+1</td>
<td>0</td>
<td>+1</td>
</tr>
</tbody>
</table>

Table 1.2: State machine description of 1 − D channel

Using this table a trellis can be drawn as shown in fig 1.12. The labels for the edges are denoted input/output.

![Trellis for 1 − D channel](image)

Figure 1.12: Trellis for 1 − D channel

From the trellis, the update equations for the Viterbi algorithm can be derived. The possible channel symbols are \{-2, 0, +2\}. Hence 3 branch metrics are required. These are\(^1\)

\[
\lambda_{-2} = -(y_k + 2)^2 \\
\lambda_0 = -y_k^2 \\
\lambda_{+2} = -(y_k - 2)^2
\]

Multiplying out and eliminating constant factors results in

\[
\lambda_{-2} = -y_k - 1 \\
\lambda_0 = 0 \\
\lambda_{+2} = y_k - 1
\]

Hence no multiplications are required.

There are two path metrics \(\Gamma_{+1}^k\) and \(\Gamma_{-1}^k\), and corresponding survivor path memories \([S_{+1}^k]\) and \([S_{-1}^k]\). These are updated as follows

\[
\Gamma_{+1}^{k+1} = Max(\Gamma_{+1}^k + \lambda_0, \Gamma_{-1}^k + \lambda_{+2}) \quad (1.22)
\]

\[
\Gamma_{-1}^{k+1} = Max(\Gamma_{-1}^k + \lambda_0, \Gamma_{+1}^k + \lambda_{-2}) \quad (1.23)
\]

If \(\Gamma_{+1}^k + \lambda_{+2}\) is chosen for \(\Gamma_{+1}^{k+1}\) then the corresponding survivor path update is \([S_{+1}^{k+1}] = [S_{+1}^k]\) otherwise \([S_{+1}^{k+1}] = [S_{+1}^k]\). If \(\Gamma_{-1}^k + \lambda_{-2}\) is chosen for \(\Gamma_{-1}^{k+1}\) then the corresponding survivor path update is \([S_{-1}^{k+1}] = [S_{-1}^k]\) otherwise \([S_{-1}^{k+1}] = [S_{-1}^k]\) where \([S_b]b\) denotes the symbol \(b\) being appended to the path \([S_b]\).

\(^1\)These are defined with a negative sign and the maximum metrics chosen.
In the case of the $1 - D$ channel, a single metric defined as the difference between $\Gamma^k_{k+1}$ and $\Gamma^k_{k-1}$ can be defined. This single metric can then be updated and stored more efficiently than described above [CDH+92].

**EPR4 Detector**

The EPR4 detector has 8 states. Fig 1.13 shows the trellis diagram. Each state is labeled as the previous 3 channel inputs. There are 5 possible channel outputs $\{-4, -2, 0, +2, +4\}$, and hence 5 branch metrics are required. The ACS updates must be performed on each of the 8 states every channel symbol. Hence the detector is 4 times more complex than the PR4 detector and must operate at twice the rate. This requires full parallel processing of each state to operate at the symbol rates of modern disk drives.

![Trellis for EPR4 channel](image)

**Figure 1.13: Trellis for EPR4 channel**

**DFE Detector**

The DFE detector requires the post cursor ISI to be calculated based on previous decisions and subtracted from the current samples. In practice, the calculation of the post cursor ISI is calculated by indexing into a RAM lookup table with a number of previous bits. Fig 1.14 shows the basic RAM-DFE configuration. In [BLMC95] a 128 word ram with 10 bits per word is used for the feedback filter. The feedback spans 7 bits. Timing and gain recovery are decision directed. Adaptation of the RAM-DFE is considered in [FCA+91].
FDTS/DF Detector

The FDTS/DF detector is similar to the DFE architecture, but requires a more complex decision element. The implementation complexity of the detector is considerable [CK91]. Analog implementations are under consideration to allow practical implementations [CBMP95] at the required data rates. To date, practical implementations have not appeared.

1.7 Conclusions

An introduction to the magnetic recording channel and present detection schemes has been presented. The main characteristics of the recording channel have been outlined. The channel is in practice a band limited binary channel with various forms of noise. The optimum detection scheme based on a linear channel model is known from communications theory. In practice, a number of compromises have to be made to allow implementation of detectors to operate at typical channel speeds. This results in a number of sub-optimum detectors that can be implemented. The basic characteristics of these schemes were outlined. At present, the PR4 PRML sampled data channel is becoming the standard in high end magnetic disk drives. The basic components of such channels such as continuous and adaptive equalization and control loops were outlined.

The EPR4 detector and DFE detectors both have recently become available as commercial products: [Ana95] and [Gra95].

The FDTS/DF detector is considered to be a possible future detector for high density recording but practical implementations have not yet appeared.

The performance of these detection systems are analyzed in chapter 2.

An alternative method of improving the performance of magnetic recording channels
is through the use of channel coding. Matched spectral null codes for this purpose are considered in chapter 3.
Chapter 2

Channel Model and Analysis

2.1 Introduction

This chapter develops an analytical model of the recording system in order to allow the evaluation of various channel architectures. The equalization required for each channel is calculated and expressions for the probability of error for the various channels are developed. The focus is on existing detection systems including the PR4 and EPR4 partial responses and the DFE and FDM/FDS/DF structures. Optimum detection through the use of the matched filter with maximum likelihood sequence detection is also calculated for comparison purposes.

A computer model of the recording channel, equalization and detectors is developed. This is used to simulate the detector and evaluate the error performance of each by counting the number of errors. These results are compared with the analytical calculations.

The structure of the model and the simulation software is general enough to allow the detectors proposed in later chapters to be analyzed and compared to the existing detectors.

2.2 System Overview

The recording detection system will normally be configured as in fig 2.1. The channel output is passed through an equalizing filter and then sampled at the baud rate for the detector.

In practice, the equalization is carried out partially in the continuous time domain with \( P(f) \), then sampled and passed through an adaptive digital filter \( F(f) \). For the purpose of analysis, the continuous time equalizer is considered to be an ideal lowpass filter which band limits the signal and noise within the Nyquist band. The constraints placed on the equalization due to implementation details are not considered and the total equalization response is denoted \( E(f) \), \( |f| \leq \frac{1}{2T} \).

The combination of the channel and equalization is normally desired to be a prechosen
response denoted $G(f)$. The choice of $G(f)$ determines the type of detector required and generally determines the complexity and performance of the system. The channel can be described as having a frequency response of $H_{\text{chan}}(f)$ and correlated noise is added to the channel output. These two factors limit the available recording density of a given channel.

![Diagram of Recording Channel and Detection System Model](image)

**Figure 2.1:** Recording channel and detection system model.

### 2.3 Channel Model

The process of recording information on magnetic media through magnetic recording is inherently a highly non-linear and complex process. However, if it is understood that the recording signal is a binary signal with saturation recording, then a linear channel model may be constructed [SW91].

![Diagram of Linearized Magnetic Recording Channel Model](image)

**Figure 2.2:** Linearized magnetic recording channel model.

Fig 2.2 shows such a linear model of the recording channel. The input signal impulse is converted to a pulse of width $T$ and unit amplitude which represents the magnetization on the media. It is assumed that the input signal is a series of bipolar impulses spaced $T$ seconds apart

$$x(t) = \sum_{k=-\infty}^{\infty} x_k \delta(t - kT) \quad x_k \in \{-1, +1\} \quad (2.1)$$
This implies that the magnetization on the medium is a bipolar signal and the non-linear effects of the recording process may be ignored\(^1\) as the medium will be saturated.

The power spectral density of the input signal will be denoted \( S_x(f) \). White Gaussian medium noise is added to the magnetization signal which is passed through the read process \( H_{\text{mag}}(f) \). White Gaussian 'electronic' noise is then added and the output signal represents the output of the channel. The total channel response to an impulse is denoted \( H_{\text{chan}}(f) \).

The Lorentz channel response is often used as a model for magnetic recording due to its use of a single parameter \( PW_{50} \) to describe the recording channel [SW91]. The step response of the Lorentz channel is

\[
h_{\text{lorz}}(t) = \frac{1}{1 + \frac{[2\pi]^2}{PW_{50}^2}}
\]  

(2.2)

It is shown in appendix B.1 that the frequency response of the Lorentz channel is

\[
H_{\text{mag}}(f) = j2\pi f \frac{PW_{50}}{2} e^{-\pi PW_{50}f}
\]  

(2.3)

In the linear model of the recording channel, an impulse \( \delta(t) \) into the channel causes a pulse of width \( T \) which can be represented as the difference of two step functions \( u(t) - u(t-T) \). The Fourier transform of this pulse is a sinc function

\[
u(t) - u(t-T) \leftrightarrow e^{-j\pi fT} \frac{\sin(\pi fT)}{\pi fT}
\]  

(2.4)

Hence, the channel model response to an impulse is

\[
H_{\text{chan}}(f) = H_{\text{mag}}(f) e^{-j\pi fT} \frac{\sin(\pi fT)}{\pi fT} = j e^{-j\pi fT} \frac{PW_{50}}{2} \sin(\pi fT) e^{-\pi PW_{50}f}
\]  

(2.5)

These frequency response expressions are only valid provided the input can be described as in eqn 2.1.

The noise at the channel output is a combination of the medium noise and white noise at the channel output. The power spectral density of this noise is

\[
N_{\text{chan}}(f) = \frac{N_m}{2} |H_{\text{mag}}(f)|^2 + \frac{N_e}{2}
\]  

(2.6)

### 2.4 Power Spectral Density of Data Source

The input signal into the linearized recording channel model was described previously as

\[
x(t) = \sum_{k=-\infty}^{\infty} x_k \delta(t - kT) \quad x_k \in \{-1, +1\}
\]  

(2.7)

\(^1\)Non-linear effects may still occur if the recording density is too high. This may be reduced through the use of write precompensation.
In order to calculate a power spectral density, a wide sense stationary signal is required. This can be formed by introducing a random time offset \( \Theta \), that is uniformly distributed on \([0, T]\) and independent of \(x_k\) [LM94]. The new signal is

\[
z(t) = x(t + \Theta) = \sum_{k=-\infty}^{\infty} x_k \delta(t + \Theta - kT)
\]  

(2.8)

The power spectral density \(S_z(f)\) of this signal can be calculated as the Fourier transform of the signal autocorrelation function \(R_z(\tau)\)

\[
S_z(f) = \int_{-\infty}^{\infty} R_z(\tau) e^{-2\pi f \tau} d\tau
\]  

(2.9)

The autocorrelation function of the signal is

\[
R_z(\tau) = E[z(t)z(t + \tau)]
\]  

(2.10)

and hence will depend on correlation in the recorded data sequence \(x_k\). If the data sequence is assumed to be random, then the values of the autocorrelation function may be calculated as

\[
R_z(\tau) = E[z(t)z(t + \tau)]
\]  

(2.11)

\[
= E \left[ \sum_n x_n \delta(t - nT + \Theta) \sum_m x_m \delta(t - mT + \Theta + \tau) \right]
\]

\[
= E \left[ \sum_n \sum_m x_n x_m \delta(t - nT + \Theta) \delta(t - mT + \Theta + \tau) \right]
\]

\[
= \sum_n \sum_m E[x_n x_m] E[\delta(t - nT + \Theta) \delta(t - mT + \Theta + \tau)]
\]

(2.12)

as \(x_k\) and \(\Theta\) are independent. The first expectation operator \(E[x_n x_m]\) is the autocorrelation of the data sequence \(R_z(m - n)\). The second expectation operator \(E[\delta(t - nT + \Theta) \delta(t - mT + \Theta + \tau)]\) is

\[
\frac{1}{T} \int_{0}^{T} \delta(t - nT + \Theta) \delta(t - mT + \Theta + \tau) d\Theta
\]  

(2.13)

Letting \(i = m - n\),

\[
R_z(\tau) = \frac{1}{T} \sum_i R_x(i) \sum_n \int_{0}^{T} \delta(t - nT + \Theta) \delta(t - (i + n)T + \Theta + \tau) d\Theta
\]  

(2.14)

Letting \(\alpha = t + \Theta - nT\)

\[
R_z(\tau) = \frac{1}{T} \sum_i R_x(i) \sum_n \int_{0}^{T-nT} \delta(\alpha) \delta(\alpha - iT + \tau) d\alpha
\]  

(2.15)

and the sum of integrals becomes a single integral

\[
R_z(\tau) = \frac{1}{T} \sum_i R_x(i) \int_{-\infty}^{\infty} \delta(\alpha) \delta(\alpha - iT + \tau) d\alpha
\]  

(2.16)
For random binary data, the data samples $x_k$ are independent and $x_k \in \{\pm 1\}$. The autocorrelation $R_x(i)$ is hence

$$R_x(i) = \begin{cases} 1 & i = 0 \\ 0 & \text{otherwise} \end{cases}$$

(2.17)

and hence

$$R_x(\tau) = \frac{1}{T} \delta(\tau)$$

(2.18)

and the data power spectral density is the Fourier transform of the autocorrelation function

$$S_z(f) = \int_{-\infty}^{\infty} \frac{1}{T} \delta(\tau) e^{-j2\pi f \tau} d\tau = \frac{1}{T}$$

(2.19)

When the data sequence to be recorded is not random then there will be correlation between the data bits, and the power spectral density of the data will be different. For example, the power spectral density of a zero disparity code of length $n$ is calculated in [Imm91] as

$$S_z(f) = \frac{1}{T} \frac{n}{n-1} \left(1 - \left(\frac{\sin(n\pi f T)}{n\sin(\pi f T)}\right)^2\right)$$

(2.20)

### 2.5 Equalization

The purpose of equalization is to select the equalizer response $E(f)$ such that the output of the channel and equalizer meet some desired response. The choice of response is considered in section 2.6.

Ideally, the equalizer output would be sampled every $T$ seconds and the original symbols would be detected from the samples. In general, the overall response from the input of the channel to the equalizer output is desired to be $G(f)$. This is chosen to meet some desirable conditions that will allow detection of transmitted data. This may be to eliminate ISI as in waveform restoration channels or to allow predefined ISI as in partial response systems. With $G(f)$ chosen, the equalizer filter may be designed according to various criteria such as zero forcing equalizers or minimum mean square error equalizers.

#### 2.5.1 Zero Forcing Equalizer

Given a desired response $G(f)$ and channel response $H(f)$, the most straightforward equalizer design can be obtained from the Zero Forcing (ZF) criteria. This determines the equalizer $E(f)$ as

$$E(f) = \frac{G(f)}{H(f)}$$

(2.21)

This requires that the equalizer completely eliminates ISI. However, this may result in considerable noise enhancement and the equalizer may not exist if the channel has a null at some frequency.
2.5.2 Minimum Mean Square Error Equalizer

More commonly, a Minimum Mean Square Error (MMSE) criteria is applied. In this case the equalizer is chosen to minimize the error in the signal taking both noise and ISI into account. The error in the output signal is the difference between the desired signal and the actual signal, and has a power spectral density of

\[ S_e(f) = S_x(f)|G(f) - H(f)|E(f)|^2 + N(f)|E(f)|^2 \]  \hspace{1cm} (2.22)

with \( S_x(f) \) being the input signal power spectral density and \( N(f) \) being the noise power spectral density. The MMSE requirement minimizes the error power given by

\[ \text{Error Power} = \int_{-\infty}^{\infty} S_e(f) df \]  \hspace{1cm} (2.23)

As \( S_e(f) \) is a power spectral density, it is non-negative. With no complexity constraint placed on the choice of the equalizer filter, the value of \( E(f) \) may be chosen independently at each desired value of \( f \). Hence, to minimize the mean square error, the value of \( S_e(f) \) should be minimized at each value of \( f \).

Consider the error in eqn 2.22. This can be written (with the functions of \( f \) implicit) as

\[
S_e = S_x(G - HE)(G - HE)^* + NEE^*
= S_x(GG^* - GH^*E^* - HEG^* + HEH^*E^*) + NEE^*
= S_xGG^* - S_xGH^*E^* - S_xHEG^* + S_xHEH^*E^* + NEE^*
= EE^*(S_xHH^* + N) - S_xGH^*E^* - S_xHG^*E + S_xGG^*
\]  \hspace{1cm} (2.24)

Defining \( S_y = S_xHH^* + N \)

\[
S_e = EE^*S_y - S_xG^*H^*E^* - S_xHG^*E + S_xGG^*
= S_y\left(EE^* - \frac{S_x}{S_y}G^*H^*E^* - \frac{S_x}{S_y}HG^*E\right) + S_xGG^*
\]  \hspace{1cm} (2.25)

Adding and subtracting \( \left(\frac{S_x}{S_y}\right)^2 GG^*HH^* \)

\[
S_e = S_y\left(EE^* - \frac{S_x}{S_y}G^*H^*E^* - \frac{S_x}{S_y}HG^*E + \left(\frac{S_x}{S_y}\right)^2 GG^*HH^* - \left(\frac{S_x}{S_y}\right)^2 GG^*HH^*\right) + S_xGG^*
= S_y\left(EE^* - \frac{S_x}{S_y}G^*H^*E^* - \frac{S_x}{S_y}HG^*E + \left(\frac{S_x}{S_y}\right)^2 GG^*HH^*\right) - S_y\left(\frac{S_x}{S_y}\right)^2 GG^*HH^* + S_xGG^*
= S_y\left(E - \frac{S_x}{S_y}G^*H^*\right)\left(E - \frac{S_x}{S_y}HG^*\right) + S_y\left(\frac{S_x}{S_y}\right)^2 GG^*(1 - S_xHH^*)
= S_y\left(E - \frac{S_x}{S_y}G^*H^*\right)^2 + S_y\left(\frac{S_x}{S_y}\right)^2 GG^*N
= S_y\left|E - \frac{S_x}{S_y}G^*H^*\right|^2 + S_y\left|G\right|^2N
\]  \hspace{1cm} (2.26)
Hence, the error power at a frequency $f$ is

$$S_e(f) = S_p(f) \left| E(f) - \frac{S_x(f)}{S_p(f)} G(f) \right|^2 \left| \frac{S_x(f)}{S_p(f)} \right|^2 N(f)$$

(2.27)

and to minimize this with respect to $E(f)$ requires setting the term $E(f) - \frac{S_x(f)}{S_p(f)} G(f) H(f)^*$ equal to zero. The required MSE equalizer is hence

$$E(f) = \frac{S_x(f) G(f) H(f)^*}{S_x(f) |H(f)|^2 + N(f)}$$

(2.28)

and substituting for $S_p(f)$

$$E(f) = \frac{S_x(f) G(f) H(f)^*}{S_x(f) |H(f)|^2 + N(f)}$$

(2.29)

The equalizer error is

$$S_e(f) = \frac{S_x(f) |G(f)|^2 N(f)}{S_x(f) |H(f)|^2 + N(f)}$$

(2.30)

This is the error signal at the equalizer output. When this is sampled at the baud rate of $\frac{1}{T}$, the resulting samples have a power spectral density of

$$S_e(e^{j2\pi ft}) = \frac{1}{T} \sum_n S_e(f - n/T)$$

(2.31)

The autocorrelation of the noise samples will then be

$$R_n(k) = T \int_{-\frac{T}{2}}^{\frac{T}{2}} S_x(e^{j2\pi ft}) e^{j2\pi fkT} df$$

(2.32)

The noise correlation of the detector samples can be used to calculate the performance of the various detection systems as developed in section 2.7. The error power spectral density in eqn 2.30 includes Gaussian noise and ISI. For the purpose of detector bit error analysis, the noise is assumed to be Gaussian.

With no noise, the MMSE equalizer reverts to a zero forcing equalizer. It can be seen that a high noise level at a particular frequency will reduce the equalizer magnitude response at that frequency. The equalizer also takes account of the power spectral density of the data source.

In a practical magnetic recording channel, the response of the channel will normally not be well known and will vary as a function of recording density. In sampled data read channels the digital equalizer is often configured to be adaptive. This allows the equalization response to track variations in the recording channel response over time. Such adaptive equalizers commonly use the LMS algorithm which converges towards the MMSE equalizer response.
2.6 Choice of Desired Response

The choice of the desired equalizer response determines the properties of the signal at the equalizer output and hence, the detection method required. Certain partial response targets have been recognized as suitable targets for the magnetic recording channel. Decision feedback type detectors have also been proposed. The following sections consider the required target responses for each.

2.6.1 Partial Response Systems

Partial response systems allow a channel response such that the output of the channel may have ISI, but that this ISI is well defined and may be accounted for in the detection process. In a partial response system, the system transfer function is forced to be a suitable polynomial in terms of $T$-second delay operators $D$. The PR4 system is defined by the response $H_{PR4}(D) = 1 - D^2$. The frequency response of this is calculated with $D = e^{-j2\pi f T}$. Hence, the PR4 response is

$$H_{PR4}(f) = 4j \sin(\pi f T) \cos(\pi f T) e^{-j2\pi f T} \quad (2.33)$$

This response has a null at DC and at the Nyquist frequency. If this response is windowed to the Nyquist frequency with

$$H_{PR4}(f) = \begin{cases} 
4j \sin(\pi f T) \cos(\pi f T) e^{-j2\pi f T} & -\frac{1}{2T} \leq f \leq -\frac{1}{2T} \\
0 & \text{otherwise}
\end{cases} \quad (2.34)$$

then it is possible to signal within the Nyquist band and still retain timing margin [KP75]. As this windowed function is continuous, it is possible to equalize to this response more easily than with a brick wall type filter.

Partial response systems were originally considered for data transmission over band limited channels. However, their applicability to magnetic recording was later observed [Kob71]. The PR4 response is considered a good choice for the magnetic recording channel as it has a null at DC similar to the magnetic recording channel. The magnetic recording channel also has severe attenuation at high frequencies and this corresponds well with the Nyquist frequency null of the PR4 response over a range of recording densities.

At higher recording densities the extended PR4 response is considered suitable. The EPR4 response is defined by $H_{EPR4}(D) = (1 - D)(1 + D)^2$ with a frequency response of

$$H_{EPR4}(f) = 8j \sin(\pi f T) \cos^2(\pi f T) e^{-j3\pi f T} \quad (2.35)$$

These responses are included in the set of partial responses of the form $(1 - D)(1 + D)^n$ which are considered appropriate in the magnetic recording channel by [TP87].
The equalization required to shape the recording channel into one of these partial response channels can be calculated using either the zero forcing (ZF) criteria or minimum mean square error (MMSE) criteria.

2.6.2 Decision Feedback Equalizer

For the decision feedback detection system the desired response is not a fixed response but is chosen as a function of the channel response. The equalization required for the DFE channel may be determined from a zero forcing or a MMSE criteria [LM94]. Fig 2.3 shows the MSE-DFE detection system.

![DFE detection system diagram](image)

Figure 2.3: DFE detection system

The channel model in section 2.3 had a transfer function of $H_{\text{chan}}(f)$ and was corrupted with noise which had a power spectral density of $N_{\text{chan}}(f)$. The MMSE decision feedback equalizer may be determined by first equalizing the channel to a flat response with an equalizer $C(f)$. This equalizer can be calculated using the MMSE criteria of eqn 2.29 with $G(f) = T$, $|f| \leq \frac{1}{2T}$

$$C(f) = \frac{TS_x(f)H_{\text{chan}}(f)^*}{S_x(f)|H_{\text{chan}}(f)|^2 + N_{\text{chan}}(f)} , \quad |f| \leq \frac{1}{2T}$$  \hspace{1cm} (2.36)

This yields a target response with the MSE minimized but with a correlated error signal. This error signal power spectral density is (from eqn 2.30)

$$S_e(f) = \frac{T^2S_x(f)}{S_x(f)H_{\text{chan}}(f)H_{\text{chan}}(f)^* + N_{\text{chan}}(f)}N_{\text{chan}}(f) , \quad |f| \leq \frac{1}{2T}$$ \hspace{1cm} (2.37)

The sampled signal has a power spectral density of

$$S_e(e^{j2\pi ft}) = \frac{1}{T}S_e(f) , \quad |f| \leq \frac{1}{2T}$$  \hspace{1cm} (2.38)

The sampled channel $H_{\text{chan}}(f)$ and equalizer $C(f)$ will have a discrete time frequency response of

$$H_{\text{df}}(e^{j2\pi ft}) = \frac{1}{T}H_{\text{chan}}(f)C(f) , \quad |f| \leq \frac{1}{2T}$$  \hspace{1cm} (2.39)
and corresponding D transform $H_{dfe}(D)$.

This sampled signal is passed through a forward filter $E(D)$. This filter is required to decorrelate the error signal $S_e(D)$ while only introducing post-cursor ISI which can be removed by the decision feedback. To meet this requirement, $E(D)$ must be monic and decorrelate the noise. Thus, $E(D)$ must satisfy

$$\frac{1}{S_e(D)} = A^2 E(D) E(D^{-1}), \quad E(D) \text{ monic and causal} \quad (2.40)$$

Thus, $E(D)$ is the minimum phase spectral factorization of $\frac{1}{S_e(D)}$ [LM94].

The target response of the MSE-DFE is hence

$$G_{dfe}(D) = H_{dfe}(D) E(D) \quad (2.41)$$

The noise at the decision point $N(D)$ has a power spectral density of $S_e(D)$ passed through the filter $E(D)$ which is

$$N(D) = S_e(D) E(D) E(D^{-1}) = \frac{1}{A^2} \quad (2.42)$$

from eqn 2.40.
Hence, the noise variance at the decision point is

$$\sigma^2_{DFE} = \frac{1}{A^2} \quad (2.43)$$

This error signal contains both ISI and Gaussian noise.

As an example, the MSE-DFE target response for the Lorentz channel at a density of 2.5 bits per $PW_{50}$ is shown in fig 2.4.

2.7 Detector Performance

Given the noise power spectral density for a particular detection target, the performance of the detector can be calculated. In this section, expressions for the error performance of the PR4 and EPR4 partial response channels and the DFE and FDTS/DF detectors are developed.

2.7.1 Threshold Detection

Threshold detectors determine the data values by comparing the received signals to predetermined levels and making decisions based on the comparison results. For a waveform restoration channel, the received signal is assumed to be a logic 1 if the signal is greater than 0 or a logic 0 if the received signal is less than 0. The probability of error for such a detector depends on the signal level and noise power at the decision point.
Consider a system with the equalizer designed to give signals of \( \pm 1 \) in the absence of noise. Let the actual signal have Gaussian Noise with a variance \( \sigma^2 \) added. In this case, a single bit error will occur when the received signal should have been \( +1.0 \) but the noise was less than \( -1.0 \) or the received signal should have been \( -1.0 \) and the noise was greater than \( +1.0 \). If the recorded data is assumed to be random, then the probability of a symbol being a 1 is \( \frac{1}{2} \) or being a 0 is \( \frac{1}{2} \). The probability of error is hence

\[
P_e = \frac{1}{2}P(n_k > 1.0) + \frac{1}{2}P(n_k < -1.0)
\]  

(2.44)

The probability that the noise sample \( n_k > 1.0 \) is the probability that a random variable with a Gaussian distribution with variance \( \sigma^2 \) is greater than 1.0, and can be written with the \( Q() \) function as

\[
P(n_k > 1.0) = Q \left( \frac{1}{\sigma} \right)
\]  

(2.45)

The value of \( P(n_k < -1.0) \) is identical due to the symmetry of the Gaussian distribution. Hence, the probability of error is

\[
P_e = Q \left( \frac{1}{\sigma} \right)
\]  

(2.46)

2.7.2 Detection for Partial Response Systems

For a partial response system, there is correlation between the signal samples in the detector due to memory in the channel. However, using precoding as in section 1.6.1, it is possible to determine the user data in a memoryless fashion from a single data sample.

Consider the case of the PR4 channel with ideal samples of \(-2, 0, +2\). When the data is precoded, user logic 1 data bits result in symbols of \( \pm 2 \) while user logic 0 data bits results
in symbols of 0. If the user data is random then the probability of a 0 channel symbol is \( \frac{1}{2} \) while the probability of a +2 and a −2 will be \( \frac{1}{4} \) and \( \frac{1}{4} \) respectively. The probability of an error in choosing the correct symbol using threshold detection levels of 1.0 and −1.0 is hence

\[
P_e = \frac{1}{2}(P(n_k > 1.0) + P(n_k < -1.0)) + \frac{1}{4}P(n_k > +1.0) + \frac{1}{4}P(n_k < -1.0)
\]  

(2.47)

which with Gaussian noise of variance \( \sigma^2 \) is

\[
P_e = \frac{3}{2}Q\left(\frac{1}{\sigma}\right)
\]  

(2.48)

Mistaking a +2 for a −2 will not result in any user bit errors but is unlikely to occur as its probability is of the order \( Q(2/\sigma) \).

However, the performance of detection on partial response channels can be much improved by considering sequences of samples and maximum likelihood detection. This is normally performed using the Viterbi algorithm. In this case the probability of error depends on the minimum distance between sequences \( d_{\text{min}} \).

\[
d_{\text{min}}^2 = \min_{\text{All seqs}} \sum_k (\hat{y}_k - \hat{y}_k)^2
\]  

(2.49)

where \( \hat{y}_k \) and \( \hat{y}_k \) are any allowable non-identical sequences. For the PR4 channel with samples −2, 0, +2, the minimum distance is

\[
d_{\text{min}} = 2\sqrt{2}
\]  

(2.50)

The probability of error in the presence of Gaussian uncorrelated noise for the PR4 channel can be calculated as

\[
P_e = KQ\left(\frac{d_{\text{min}}}{2\sigma}\right)
\]  

(2.51)

with \( K \) a constant depending on the number of errors per error event and the probability of the error event being supported. For the PR4 channel the probability of error is[WP86]

\[
P_e = 4Q\left(\frac{2\sqrt{2}}{2\sigma}\right)
\]  

(2.52)

In this case the expression in the argument of the \( Q() \) function is \( \sqrt{2} \) times greater than the case of threshold detection. Ignoring the constant term, this implies that the Viterbi detector can achieve the same error rate as a threshold detector but with a noise power that is 2 times as large i.e., 3dB additional noise.

However, at higher recording densities, the equalization required to shape the recording channel to the desired partial response channel can introduce considerable correlation in the noise samples received by the Viterbi detector. In this case, the detector performance depends on the noise correlation.
2.7.3 Performance of Viterbi Detector with Noise Correlation

In general for an ISI channel with inputs \( x_k \), the channel output \( y_k \) can be written as

\[
y_k = \sum_{i=0}^{N-1} h_i x_{k-i} + n_k
\]  

(2.53)

with \( h_k \) being the discrete time channel response and \( n_k \) being the noise samples. In the absence of noise, the channel output is the sequence \( \hat{y}_k \). The actual received signal is \( y_k \).

Let \( \tilde{y}_k \) be an output sequence different from the correct sequence \( \hat{y}_k \). Thus

\[
\hat{y}_k = \sum_{i=0}^{N-1} h_i x_{k-i} \quad \text{Correct Sequence} \tag{2.54}
\]

\[
\tilde{y}_k = \sum_{i=0}^{N-1} h_i \tilde{x}_{k-i} \quad \text{Incorrect Sequence} \tag{2.55}
\]

\[
y_k = \sum_{i=0}^{N-1} h_i x_{k-i} + n_k \quad \text{Received Sequence} \tag{2.56}
\]

The channel can be considered to be a state machine with the states \( s_k \) denoted by the sequence of \( N - 1 \) input symbols \( \{x_1, x_2, \ldots, x_{N-1}\} \). An error event occurs when the detected state \( \hat{s}_k \) is different from the correct state \( \tilde{s}_k \). The error event lasts from the time the correct state and the detected state diverge at index \( k_1 \) to the time that they first remerge at index \( k_2 \). Hence, for an error event to occur, the error sequence of channel outputs \( \tilde{y}_k \) is mistaken for the correct sequence \( \hat{y}_k \) due to the noise.

With maximum likelihood detection, this will occur when

\[
\sum_{k=k_1}^{k_2} (\hat{y}_k - y_k)^2 < \sum_{k=k_1}^{k_2} (\tilde{y}_k - y_k)^2
\]  

(2.57)

However, \( \hat{y}_k + n_k = y_k \) and so

\[
\sum_{k=k_1}^{k_2} (\hat{y}_k - \tilde{y}_k - n_k)^2 < \sum_{k=k_1}^{k_2} (-n_k)^2
\]  

(2.58)

Define the channel error sequence \( e_k = \hat{y}_k - \tilde{y}_k \) and so

\[
\sum_{k=k_1}^{k_2} (e_k - n_k)^2 < \sum_{k=k_1}^{k_2} (n_k)^2
\]  

(2.59)

and hence, an error event occurs when

\[
\sum_{k=k_1}^{k_2} \frac{e_k^2}{2} < \sum_{k=k_1}^{k_2} e_k n_k
\]  

(2.60)

The right hand side term is a random variable, say \( X \) with

\[
X = \sum_{k=k_1}^{k_2} e_k n_k
\]  

(2.61)
Without loss of generality let \( k1 = 0 \) and \( k2 = J \). The variance of \( X \) is

\[
\text{Var}[X] = E[X^2] = E \left( e_0^2 n_0^2 + e_0 n_0 e_1 n_1 + \ldots + e_0 n_0 e_J n_J + e_1^2 n_1^2 + \ldots + e_1 n_1 e_J n_J + \ldots + e_J^2 e_J n_J \right)
\]

\[
= E[e_0^2 n_0^2] + E[e_0 n_0 e_1 n_1] + \ldots + E[e_J^2 e_J^2] = \sum_{k=0}^{J} e_k^2 E[n_k^2] + \sum_{i=0}^{J} \sum_{j=0, j \neq i}^{J} e_i e_j E[n_i n_j]
\]

\[
= \sum_{k=0}^{J} e_k^2 R_n(0) + \sum_{i=0}^{J} \sum_{j=0, j \neq i}^{J} e_i e_j R_n(i - j)
\]

(2.62)

where \( R_n(\tau) \) is the autocorrelation of the noise. The probability of an error event occurring is the probability that a random variable \( X \) is greater than \( \sum_{k=0}^{J} e_k^2 \) as in eqn 2.60. The probability of this occurring is

\[
Q\left( \frac{\sum_{k=0}^{J} e_k^2}{\sqrt{\text{Var}[X]}} \right)
\]

(2.63)

Hence, the probability of an error event is

\[
P(\text{Error Event}) = Q\left( \frac{\sum_{k=0}^{J} e_k^2}{\sqrt{\sum_{k=0}^{J} e_k^2 R_n(0) + \sum_{i=0}^{J} \sum_{j=0, j \neq i}^{J} e_i e_j R_n(i - j)}} \right)
\]

(2.64)

The probability of a bit error can then be calculated as

\[
P(\text{Bit Error}) = \sum_{\text{All Error Events}} P(\text{Event Occurs}) N_{\text{Errors per Event}} P(\text{Error Event})
\]

(2.65)

If the noise is uncorrelated, then eqn 2.64 simplifies to

\[
Q\left( \frac{\sqrt{\sum_{k=0}^{J} e_k^2}}{2\sigma_n} \right)
\]

(2.66)

where \( \sigma_n^2 = R_n(0) \) and the term \( \sqrt{\sum_{k=0}^{J} e_k^2} \) is the distance between the correct event and the error event. The error rate will be dominated by the minimum distance error event and so a common expression for the bit error rate with uncorrelated noise is

\[
P_{\text{error}} \approx K Q\left( \frac{d_{\text{min}}}{2\sigma_n} \right)
\]

(2.67)

with \( K \) a small constant.
2.7.4 Performance of EPR4 Viterbi Detector

In the case of an EPR4 detector, the channel response is \( H(D) = 1 + D - D^2 - D^3 \). Hence, the detected output \( \tilde{y}_k \) and the correct samples \( \hat{y}_k \) are

\[
\tilde{y}_k = \hat{x}_k + \hat{x}_{k-1} - \hat{x}_{k-2} - \hat{x}_{k-3} \\
\hat{y}_k = \hat{x}_k + \hat{x}_{k-1} - \hat{x}_{k-2} - \hat{x}_{k-3}
\]  

(2.68) \hspace{1cm} (2.69)

Define the input error sequence as

\[
a_k \equiv \hat{x}_k - \hat{x}_k
\]  

(2.70)

Hence, \( a_k \in \{-2, 0, +2\} \) and the error event sequence \( e_k \) is

\[
e_k = a_k + a_{k-1} - a_{k-2} - a_{k-3}
\]  

(2.71)

Consider an error event starting at time \( u \) and ending at time \( v \). Hence, for \( k < u \) \( \hat{x}_{k-1} = \hat{x}_{k-1} = 0 \). At time \( k = u \) \( \hat{x}_{k-1} \neq \hat{x}_{k-1} \Rightarrow a_{k-1} \neq 0 \)

Similarly, at the end of the error event with time \( k = v \) \( \hat{x}_{k-3} \neq \hat{x}_{k-3} \Rightarrow a_{k-3} \neq 0 \) and after the error event \( k > v \) \( \hat{x}_k = \hat{x}_k \Rightarrow a_k = 0 \). During the error event there must exist some \( a_k \neq 0 \), otherwise, two separate error events should be considered. This information can be summarized in table\(^2\) 2.1.

<table>
<thead>
<tr>
<th>( k )</th>
<th>u-2</th>
<th>u-1</th>
<th>u</th>
<th>u+1</th>
<th>u+2</th>
<th>...</th>
<th>v-3</th>
<th>v-2</th>
<th>v-1</th>
<th>v</th>
<th>v+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_k )</td>
<td>+2</td>
<td>0,±2</td>
<td>0,±2</td>
<td>0,±2</td>
<td>At least</td>
<td>±2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( a_{k-1} )</td>
<td>0</td>
<td>+2</td>
<td>0,±2</td>
<td>0,±2</td>
<td>one non</td>
<td>±2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( a_{k-2} )</td>
<td>0</td>
<td>0</td>
<td>±2</td>
<td>0,±2</td>
<td>zero</td>
<td>0,±2</td>
<td>0,±2</td>
<td>±2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( a_{k-3} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>±2</td>
<td>entry</td>
<td>0,±2</td>
<td>0,±2</td>
<td>0,±2</td>
<td>±2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.1: Error event starting at \( k = u \) and ending at \( k = v \).

Only minimum distance error events are of interest and the minimum distance error events have a distance of 4. From table 2.1, it can be seen that error events must be at least of length 4. In order to take account of noise correlation, the nature of the minimum distance error events must be described. This can be done by filling table 2.1 with all possible combinations and noting the minimum distance events. Table 2.2 shows a minimum distance event of length 4 with \( e_k \) calculated as \( a_k + a_{k-1} - a_{k-2} - a_{k-3} \). The error event shown is \( e_k = \{+2, +2, -2, -2\} \). Only one detected bit error occurs for this error event. The negative of this \( e_k \) is also an error event. These are the only two error events of length 4 with minimum distance. There are 16 possible data sequences of length 4 and

\(^2\)The columns in the table may overlap e.g. when \( u - 1 = v - 3 \)
Table 2.2: Error event of length 4.

<table>
<thead>
<tr>
<th>$a_k$</th>
<th>+2</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{k-1}$</td>
<td>0</td>
<td>+2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a_{k-2}$</td>
<td>0</td>
<td>0</td>
<td>+2</td>
<td>0</td>
</tr>
<tr>
<td>$a_{k-3}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+2</td>
</tr>
<tr>
<td>$e_k$</td>
<td>+2</td>
<td>+2</td>
<td>-2</td>
<td>-2</td>
</tr>
</tbody>
</table>

This can be continued for increasing lengths and the pattern observed. Events of length 4 were described above. For even events of length $l$ there are two types of error event with the following properties

Type 1

$$P(\text{Error Event Supported}) = \frac{2^{(2+l/2)}}{2^l}$$

$$N_{\text{Error Per Event}} = \frac{l}{2} - 1$$

$$e_k = \pm\{+2, +2, 0, \ldots, 0, -2, -2\} \quad (2.75)$$
and type 2

\[
P(\text{Error Event Supported}) = \frac{16}{2^l} \\
N_{\text{Error Per Event}} = l - 3 \\
e_k = \pm\{+2, 0, -2, \ldots, 0, +2, 0, -2\} \quad (2.76)
\]

For odd lengths \( l \) only one type of minimum distance error events can occur and is described as

\[
P(\text{Error Event Supported}) = \frac{16}{2^l} \\
N_{\text{Error Per Event}} = l - 3 \\
e_k = \pm\{+2, 0, -2, \ldots, 0, -2, 0, +2\} \quad (2.77)
\]

Using these properties and the expressions for the performance of a Viterbi detector in the presence of correlated noise given by eqn 2.65 and eqn 2.64, the following expression for the bit error rate of an EPR4 Viterbi detector was calculated as (appendix B.2)

\[
P_e(\text{EPR4}) = Q\left(\frac{2}{\sqrt{R_n(0) + R_n(1)/2 - R_n(2) - R_n(3)/2}}\right) \\
+ \sum_{l=0}^{\infty} \frac{16}{2^l(l - 3)}Q\left(\frac{2}{\sqrt{R_n(0) - R_n(2) - R_n(l - 3) + R_n(l - 5)/2 + R_n(l - 1)/2}}\right) \\
+ \sum_{l=0}^{\infty} \frac{4}{2^l/2(l/2 - 1)}Q\left(\frac{2}{\sqrt{R_n(0) + R_n(1) - R_n(l - 2) - R_n(l - 3)/2 - R_n(l - 1)/2}}\right) \\
+ \frac{16}{2^l(l - 3)}Q\left(\frac{2}{\sqrt{R_n(0) - R_n(2) + R_n(l - 3) - R_n(l - 5)/2 - R_n(l - 1)/2}}\right) \\
(2.78)
\]

If there is no noise correlation, this reduces to

\[
P_e(\text{EPR4}) = 6Q\left(\frac{2}{\sigma_n}\right) \quad (2.79)
\]

with \( \sigma_n^2 = R_n(0) \)

Simulation Verification

In order to verify the above expression, a computer simulation of an EPR4 Viterbi detector with correlated noise was carried out. An EPR4 sequence was generated from random data. White Gaussian noise samples were passed through a FIR filter with the weights \( \{0.409892, 0.347709, 0.289306, 0.165171, 0.100000\} \). This results in noise with autocorrelation function \( R_n(k) \approx \{0.04, 0.1, 0.2, 0.3, 0.4, 0.3, 0.2, 0.1, 0.04\} \). The simulations were run
until 100 errors occurred. Fig 2.5 shows the results. For error rates corresponding to effective SNRs above 10.5dB, there is good agreement with the calculated values accounting for noise correlation. The calculated error rate without noise correlation is considerably in error.

**Figure 2.5: Simulated and calculated performance of EPR4 with correlated noise**

### 2.7.5 Performance of PR4 Viterbi Detector

The PR4 detector is normally configured as 2 interleaved Dicode detectors. The performance of the Dicode detector will hence be considered. Fig 2.6 shows the possible events of length 2 supported by the Dicode channel starting from state +1.

**Figure 2.6: Events of length 2 starting from state +1**

From state +1 there are 2 paths to state +1 of length 2 and both support 1 minimum distance error event. The corresponding error sequence is $e_k = \pm\{+2, -2\}$ and each results in a single bit error.

The same is true for events starting is state +1 and ending in state -1.

By symmetry, the same is true for events starting in state -1. Hence, there are 8 possible paths, each supporting 1 minimum error event.

Minimum distance events of length 2 can hence be summarized as in table 2.3.

Fig 2.7 shows the possible events of length 3 supported by the Dicode channel starting
from state +1 and ending in state +1.

![Diagram](image)

Figure 2.7: Events of length 3 starting from state +1

From state +1 there are 4 paths to state +1 of length 3 and 2 support 1 minimum distance error event of length 3. The corresponding error sequence is \( e_k = \pm\{+2, 0, -2\} \) and each results in a double bit error. The same is true for events starting in state +1 and ending in state -1. By symmetry, the same is true for events starting in state -1. Hence, there are 16 possible paths, 8 supporting 1 minimum error event of length 3.

Minimum distance events of length 3 can hence be summarized as in table 2.4.

<table>
<thead>
<tr>
<th>Length</th>
<th>( P(\text{event supported}) )</th>
<th>Number of Bit Errors</th>
<th>( e_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>( \frac{8}{9^n} )</td>
<td>2</td>
<td>( \pm{+2, 0, -2} )</td>
</tr>
</tbody>
</table>

Table 2.4: Minimum distance error events of length 3

In general, minimum distance error events of length \( l \) may be described as in table 2.5.

<table>
<thead>
<tr>
<th>Length</th>
<th>( P(\text{event supported}) )</th>
<th>Number of Bit Errors</th>
<th>( e_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l )</td>
<td>( \frac{8}{9^{l-1}} )</td>
<td>( l-1 )</td>
<td>( \pm{+2, 0, \ldots, 0, -2} ) ( \mod l-2 )</td>
</tr>
</tbody>
</table>

Table 2.5: Minimum distance error events of length \( l \)

Error events of length \( l \) have the form \( \pm\{+2, 0, \ldots, 0, -2\} \) and hence

\[
\sum_{k} e_k^2 = 8
\]

(2.80)

and

\[
\sum_{k} e_k^2 R_n(0) + \sum_{i \neq j} e_i e_j R_n(i-j) = 8 R_n(0) - 8 R_n(l-1)
\]

(2.81)
Letting $i = l - 1$ the probability of error for the Dicode channel may hence be calculated as

$$P_e(DICODE) = \sum_{i=1}^{\infty} \frac{8i}{2^{i+2}} Q \left( \frac{\sqrt{2}}{\sqrt{R_n(0) - R_n(i)}} \right)$$  \hspace{1cm} (2.82)

As the PR4 detector is two interleaved Dicode detectors, the probability of error can be determined by modifying the correlation terms to correspond to samples spaced twice as far apart.

$$P_e(PR4) = \sum_{i=1}^{\infty} \frac{2i}{2^i} Q \left( \frac{\sqrt{2}}{\sqrt{R_n(0) - R_n(2i)}} \right)$$  \hspace{1cm} (2.83)

### 2.7.6 Decision Feedback Detection

With the decision feedback target response as defined in section 2.6.2, the response of the channel to a unit impulse is a sample of value 1 followed by the tail of ISI samples. If the previous decisions were correct, the tail ISI is completely canceled. In this case the probability of making an error is the probability that the detector noise causes a sample value of +1 to be misdetected as a sample value of −1 or vice-versa.

In section 2.6.2, the noise variance at the decision point $\sigma_{DFE}^2$ was calculated in eqn 2.43. While the error signal contained both Gaussian noise and ISI, it will be assumed for the purpose of calculating error rates that the error signal is Gaussian. Hence, the probability of error for the DFE is

$$P_e(DFE) = \frac{1}{2} Q \left( \frac{1}{\sigma} \right) + \frac{1}{2} Q \left( \frac{1}{\sigma} \right)$$

$$= Q \left( \frac{1}{\sigma_{DFE}} \right)$$  \hspace{1cm} (2.84)

This expression ignores the effect of error propagation in the detector but gives an expression for the probability of an error event occurring.

### DFE Error Propagation

The effect of error propagation may be considered by modeling the error propagation as a Markov process [HS95]. Let the target response for the channel and forward equalizer be described by the time domain samples $\{h_0, \ldots, h_N\}$ with $h_0 = 1$. The channel input bits are $x_k$ and the detected bits are $\hat{x}_k$. The feedback filter will subtract the post-cursor ISI. The input to the decision element will be

$$u_k = \sum_{i=0}^{N} h_i x_{k-i} + n_k - \sum_{i=1}^{N} h_i \hat{x}_{k-i}$$  \hspace{1cm} (2.85)

where $n_k$ is the noise, assumed to be zero mean Gaussian with a variance of $\sigma_{DFE}^2$. Defining the error sequence as $e_k = x_k - \hat{x}_k$ and noting that $h_0 = 1$, the input to the decision element
is
\[ u_k = x_k + n_k + \sum_{i=1}^{N} e_{k-i} h_i \] (2.86)

In the presence of errors, the input to the decision element is corrupted with \( \sum_{i=1}^{N} e_{k-i} h_i \), possibly increasing the probability of further errors.

The state of the system at time \( k \) can be defined as a state vector \( \Xi_k \) defined as
\[ \Xi_k = \{ e_{k-1}, \ldots, e_{k-N} \} \] (2.87)

and this causes the signal \( Z(\Xi_k) = \sum_{i=1}^{N} e_{k-i} h_i \) to be added to the decision signal. Each element \( e_k \in \{-2, 0, +2\} \), and hence, there are \( 3^N \) possible state vectors. The Markov process models the probability of each state and the transitional probabilities from one state to another. Let the probability of the \( i^{th} \) state at time \( k \) be denoted \( p^i_k \). Consider the state \( \Xi_k^i = \{ e_{k-1}^i, \ldots, e_{k-N}^i \} \). At time \( k+1 \), this becomes the state \( \{ e_k, e_{k-1}^i, \ldots, e_{k-N+1}^i \} \) where \( e_k \) may take on 3 possible values \( \{-2, 0, +2\} \) corresponding to an error where \( x_k \) was \(+1\), no error and an error where \( x_k \) was \(-1\) respectively. The probabilities of each of these will depend on the feedback error signal \( Z(\Xi_k^i) \). Given \( \Xi_k^i \), the probability that the next state is \( \Xi_k = \{ 2, e_{k-1}^i, \ldots, e_{k-N+1}^i \} \) is
\[ P(\Xi_k^i | \Xi_k^j) = P(x_k = 1) P(\hat{x}_k = -1 | \Xi_k) \]
\[ = \frac{1}{2} P(1 + n_k + Z(\Xi_k^i) < 0) \]
\[ = \frac{1}{2} Q \left( \frac{1 + Z(\Xi_k^i)}{\sigma_{DFE}} \right) \] (2.88)

The probability that \( \Xi_k = \{ -2, e_{k-1}^i, \ldots, e_{k-N+1}^i \} \) is
\[ P(\Xi_k^i | \Xi_k^i) = P(x_k = -1) P(\hat{x}_k = +1 | \Xi_k) \]
\[ = \frac{1}{2} P(-1 + n_k + Z(\Xi_k^i) > 0) \]
\[ = \frac{1}{2} Q \left( \frac{1 - Z(\Xi_k^i)}{\sigma_{DFE}} \right) \] (2.89)

The only other possibility is of no error and \( \Xi_k^m = \{ 0, e_{k-1}^i, \ldots, e_{k-N+1}^i \} \) which has a probability of
\[ P(\Xi_k^m | \Xi_k^i) = 1 - P(\Xi_k^i | \Xi_k^i) - P(\Xi_k^i | \Xi_k^i) \] (2.90)

Thus, the probabilities at time \( k + 1 \) for each of the \( M = 3^N \) possible states may be updated from the probabilities of each state at time \( k \). This can be expressed as the matrix multiplication
\[
\begin{bmatrix}
  p^1_{k+1} \\
  p^2_{k+1} \\
  \vdots \\
  p^M_{k+1}
\end{bmatrix} =
\begin{bmatrix}
  P(\Xi^1 | \Xi^1) & \ldots & P(\Xi^1 | \Xi^M) \\
  P(\Xi^2 | \Xi^1) & \ldots & P(\Xi^2 | \Xi^M) \\
  \vdots & \ddots & \vdots \\
  P(\Xi^M | \Xi^1) & \ldots & P(\Xi^M | \Xi^M)
\end{bmatrix}
\begin{bmatrix}
  p^1_k \\
  p^2_k \\
  \vdots \\
  p^M_k
\end{bmatrix}
\] (2.91)
The transition probability matrix is a sparse matrix and each row and column will have only 3 non-zero elements as a given state may only extend into 3 possible states.

As the system must be in one state at any time, \( \sum_{i=1}^{M} p_{k}^{i} = 1 \). By initializing \( p_{k}^{i} = 1/M \), the matrix multiplication in eqn 2.91 may be iterated until the state probabilities \( p_{i} \) reach their steady state value. That this is guaranteed to converge is shown in [HS95].

The probability of a bit error for the DFE can then be calculated as

\[
P_e(DFE) = 1 - \sum_{i=1}^{M} p_{i}^{t}(\text{No error}|\Xi)
\] (2.92)

2.7.7 FDTS/DF Detector

The finite delay tree search with decision feedback (FDTS/DF) detector has been proposed as a suitable detector for use in high density magnetic recording channels due to its performance [MC90] and complexity on \((1,k)\) run length constrained channels [Ken91]. The detector is based on equalizing the channel as in the case of a DFE and using decision feedback to cancel the trailing ISI with the exception of a number of terms. A DFE cancels all but the first term. The FDTS/DF detector may use two or more terms. The data is detected by using a tree search on the uncanceled terms.

The potential advantage of this detector is that more of the channel’s pulse energy is used in the detection process and hence, it should perform better than a DFE. For example, considering the minimum phase response shown in fig 2.4, it can be seen that the second term of 0.6 is significant compared to the first term of 1.0.

Increasing the number of samples taken into account increases the complexity of the detector. The two sample detector is considered here.

In the two-sample case, the pulse response of the channel with the decision feedback term removed is

\[
H_{FDTS_{2}}(D) = 1 + gD
\] (2.93)

with \( g \) being the second term. This signal is corrupted by the same white noise \( \sigma_{DFE} \) as in the case of the DFE. Fig 2.8 shows the tree search for the 2 sample response.

At time index \( k + 1 \), a decision is to be made on what the input sample \( x_{k} \) should be. At this point there are 4 possible noiseless output values of \( \hat{y}_{k+1} \in \{1+g, -1+g, 1-g, -1-g\} \). These correspond to the set of input sequences \( \hat{x}_{k} \hat{x}_{k+1} \in \{+1+1, +1-1, -1+1, -1-1\} \). A branch metric of \( \lambda_{k+1} = -(y - \hat{y}_{k+1})^{2} \) is calculated for each of the possible outputs at time \( k + 1 \). Path metrics for each of the 4 nodes are calculated by adding the branch metric to the path metric of the previous node \( \Gamma_{k+1}^{-1} \) or \( \Gamma_{k}^{-1} \). This results in 4 new path metrics \( \Gamma_{k+1}^{q}, \Gamma_{k+1}^{r}, \Gamma_{k+1}^{c}, \Gamma_{k+1}^{d} \).

The maximum of these path metrics is then selected. If \( \Gamma_{k+1}^{q} \) or \( \Gamma_{k+1}^{r} \) is the maximum then \( \hat{x}_{k} \) is decided on as a +1. The metric \( \Gamma_{k+1}^{-1} \) is updated from \( \Gamma_{k+1}^{q} \) and the metric \( \Gamma_{k+1}^{-1} \)
is updated from $\Gamma^b_{k+1}$. Otherwise $\hat{x}_k$ is decided on as a $-1$ and the metric $\Gamma^+_{k+1}$ is updated from $\Gamma^c_{k+1}$ and the metric $\Gamma^-_{k+1}$ is updated from $\Gamma^d_{k+1}$. At this point the unchosen half of the tree is discarded.

A decision error occurs when a path in the upper branch is chosen in error over a path in the lower branch. The probability of this occurring depends on the minimum distance between any path in the top and any path in the bottom.

The possible sequences in the top half are

$$1 + g\hat{x}_{k-1}, 1 + g$$
and

$$1 + g\hat{x}_{k-1}, -1 + g$$

(2.94)

and possible sequences in the bottom half of the tree are

$$-1 + g\hat{x}_{k-1}, 1 - g$$
and

$$-1 + g\hat{x}_{k-1}, -1 - g$$

(2.95)

Hence, the minimum distance between both halves of the tree is

$$d_{min} = \sqrt{2^2 + \min\{(2g)^2, (2 + 2g)^2, (-2 + 2g)^2\}}$$

(2.96)

and the probability of error for the detector is

$$P_e(\text{FDTS/DF}) = Q \left( \frac{d_{min}}{2\sigma_{DFE}} \right)$$

(2.97)

As in the case of the DFE detector, the effect of error propagation may be considered through the use of a Markov model based on an error vector. The analysis is similar to that of the DFE in section 2.7.6. In this case the feedback error signal will be

$$Z(\Xi_k) = \sum_{i=1}^{N-1} e_{k-i}h_{i+1}$$

(2.98)
The transition probabilities also need to be appropriately calculated. Given $\Xi_k$, the probability that $\Xi_k = \{2, e_k^{i-1}, \ldots, e_k^{i-N+2}\}$ is

$$P(\Xi_k | \Xi_k) = P(x_k = 1)P(\bar{x}_k = -1 | \Xi_k)$$

(2.99)

The probability of error will depend on the distances between possible sequences. With $x_k = 1$ the correct output sequences will be $\{1, 1+g\}$ or $\{1, 1-g\}$ depending on the future bit $x_{k+1}$. The error sequences will be $\{1, -1+g+Z(\Xi_k)\}$ or $\{1, -1-g+Z(\Xi_k)\}$. Defining $d_{min}$ as the minimum distance between either of the correct sequences and either of the error sequences, the transition probability may be calculated as

$$P(\Xi_k | \Xi_k) = \frac{1}{2} Q\left(\frac{d_{min}}{2\sigma_{DFE}}\right)$$

(2.100)

The rest of the analysis proceeds with the iterative solution of the Markov process as in section 2.7.6.

### 2.8 Matched Filter with MLSD

The performance of the matched filter with Maximum Likelihood Sequence Detection (MLSD) may calculated to determine an upper bound for the detection performance of any system [For72]. The magnetic recording channel has a response of $H_{chan}(f)$. Correlated noise with a power spectral density $N_{chan}(f)$ is added to the channel output. Let the channel output be passed through a filter with a response $\frac{1}{\sqrt{N_{chan}(f)}}$. The output of this filter is a signal corrupted by white noise with a two sided power spectral density of 1. The response of the channel and filter is $H(f)$ with

$$H(f) = \frac{H_{chan}(f)}{\sqrt{N_{chan}(f)}}$$

(2.101)

Fig 2.9 shows the requirements for an optimum detection system for the channel $H(f)$ with additive Gaussian white noise.

The corresponding matched filter is then $H^*(f)$ [LM94]. The response of the channel, noise whitening filter and matched filter is then sampled at the symbol rate. The noise at the output of the sampler has a power spectral density of

$$S_1(e^{j2\pi fT}) = \frac{1}{T}S_n(f) \sum_k |H(f - \frac{k}{T})|^2$$

(2.102)

In order to whiten this noise, let the whitening filter $W(f)$ be defined as

$$W(e^{j2\pi fT}) = \frac{1}{\sqrt{S_1(e^{j2\pi fT})}}$$

(2.103)

51
The noise at output of this filter will have a power spectral density of 1. The sampled response of the channel $H(f)$ and the matched filter $H^*(f)$ is

$$H(f) = \frac{1}{T} \sum_k H(f - k/T) H^*(f - k/T)$$

$$= \frac{1}{T} \sum_k |H(f - k/T)|^2$$

(2.104)

The whitened matched filter will have a frequency response of

$$WMF(e^{j2\pi ft}) = \frac{1}{T} \sum_k |H(f - k/T)|^2$$

$$= \frac{1}{T} \sum_k |H(f - k/T)|^2$$

$$= \sqrt{\frac{1}{T} \sum_k |H(f - k/T)|^2}$$

$$= \frac{1}{T} \sum_k |H(f - k/T)|^2$$

(2.105)

as $S_n(f) = 1$.

The sequence into the MLSD detector from a single input impulse is

$$wmf_k = T \int_{-T/2}^{T/2} WMF(e^{j2\pi ft}) e^{j2\pi ft k} df$$

(2.106)

The noise corrupting the sequence into the MLSD detector is white and has a variance

$$\sigma_{MLSD}^2 = T \int_{-T/2}^{T/2} 1 \, df$$

$$= 1$$

(2.107)

The performance of the MLSD will be governed by the minimum distance of all allowed distinct sequences. Let $\{x_k\}$ and $\{\tilde{x}_k\}$ be two distinct sequences at the channel input. The whitened matched filter response will be truncated when the terms become small and the
response will be considered a finite sequence \(h_0, \ldots, h_{L-1}\) with \(L\) terms. The corresponding outputs from the whitened matched filter are

\[
\hat{y}_k = \sum_{i=0}^{L-1} \hat{x}_{k-i}h_i
\]  

(2.108)

and

\[
\hat{y}_k = \sum_{i=0}^{L-1} \hat{x}_{k-i}h_i
\]  

(2.109)

The output error \(e_k\) sequence can be calculated as

\[
e_k = \hat{y}_k - \hat{y}_k = \sum_{i=0}^{L-1} x_{k-i}h_i - \sum_{i=0}^{L-1} \hat{x}_{k-i}h_i
\]

\[
= \sum_{i=0}^{L-1} (\hat{x}_{k-i} - x_{k-i})h_i
\]

\[
= \sum_{i=0}^{L-1} a_{k-i}h_i
\]  

(2.110)

where \(a_k = \hat{x}_{k-i} - x_{k-i}\), the input error sequence. As \(x_k \in \{\pm 1\}\), the input error sequence has \(a_k \in \{0, \pm 2\}\). The Euclidean distance \(d\) between two output sequences is

\[
d^2 = \sum_{k} e_k^2 = \sum_{k} \left[ \sum_{i=0}^{L-1} a_{k-i}h_i \right]^2
\]  

(2.111)

The minimum distance \(d_{min}\) is the smallest distance over all possible sequences. While an exhaustive search for \(d_{min}\) is impossible an upper bound can be obtained. For all possible input error sequences of length \(N\), the minimum distance can be calculated. The minimum distance including any longer sequences must be less than or equal to the calculated minimum distance and so

\[
d_{min}^2 \leq \text{Min} \left\{ \sum_{k=0}^{N-1} \left[ \sum_{i=0}^{L-1} a_{k-i}h_i \right]^2 \right\}
\]  

(2.112)

If a single data bit is sent, \(a_k \neq 0\) for a single value of \(k\) and the minimum distance in this case is known as the matched filter bound[CM86]. This represents the performance of the detection system when ISI is not present. The matched filter bound distance is

\[
d_{mf}^2 = \sum_{i=0}^{L-1} (2h_i)^2
\]  

(2.113)

The error rate performance is determined by the minimum distance. The matched filter bound has an error rate performance governed by

\[
P_e(MFB) = Q \left( \frac{d_{mf}}{2\sigma_{MLSD}} \right)
\]  

(2.114)
while the MLSD error rate performance is governed by

\[ P_e(MLSD) = Q\left(\frac{d_{\text{min}}}{2\sigma_{MLSD}}\right) \] (2.115)

The matched filter bound performance is considered an upper bound on the performance of any possible detection system. The MLSD error rate performance is a tighter bound with the difference between the two showing the inherent loss due to ISI in the channel.

2.9 Computer Model

A software based model of the recording channel and various detection systems has been developed to allow simulation of different detection systems to be performed. Fig 2.10 shows a diagram of the model implemented. Appendix C describes further details of the software.

![Diagram](image-url)

**Figure 2.10: Computer Model Block Diagram**

Random data which may be coded as desired is over sampled. Due to the high frequency attenuation of the magnetic recording channel at high densities, an over sampling ratio of 2 was typically used but higher values may be accommodated. The model allows media noise and white noise and implements the Lorentz channel step response as described in section 2.3. The channel model output is passed through a sampled data model of a continuous time filter based on the bilinear transform. This allows modeling of various filters.
such as linear phase low pass filters with high frequency boost as discussed in section 1.6.3. The signal is then optionally low pass filtered with a 51 tap sinc FIR filter and sampled at the symbol rate of $1/T$.

The original data bits are delayed to account for the delay in the channel and passed through a target response filter. The symbol samples are passed through a LMS adaptive filter. The LMS filter outputs are subtracted from the ideal signals generated by the target response to create an error signal for the adaption algorithm.

The output samples from the LMS filter are also sent to the detector which detects the data for the target response selected. The output of the detector is compared with suitably delayed data from the data source and the number of errors are counted. In this way, the bit error rate performance of the detector can be counted. The accuracy of the error rate depends on the number of errors counted and the actual error rate. The limits for a 95% confidence interval in the measured error rate as a function of error rate and number of errors counted are considered in appendix D.

In this thesis, simulated error rates with the computer model are calculated for simulations with 100 bit errors and an error rate of less than $1 \times 10^{-5}$. This error rate corresponds to an effective SNR of 12.6dB. The 95% confidence interval with 100 errors counted is less than $\pm 0.1$dB.

For DFE type structures, such as the DFE and FDTS/DF detectors, an optional feedback equalizer based on the detected data may be used. This can be adapted based on the LMS error signal.

### 2.10 Calculations and Simulations

Using the noise autocorrelation of the detector samples as in section 2.5.2 and the detector error performance developed in section 2.7, the error rate performance of various channels can be calculated. These channels have also been simulated using the computer model. The following plots show the calculated and simulated performance of the various detectors. The channel SNR is defined based on the electronics noise $N_e$ (fig 2.2) as³

$$\text{Channel SNR} = 10 \log_{10} \left( \frac{4}{N_e} \right)$$

(2.116)

#### 2.10.1 Electronics Noise

Fig 2.11(a) shows the calculated and simulated required channel SNR for the PR4 channel to achieve a bit error rate of $10^{-6}$ over a range of recording densities from 1.5 to 3.5 bits per $PW_{50}$ considering electronics noise only. Fig 2.11(b) shows the EPR4 channel

³The relative difference between SNR's is of interest
results, fig 2.12(a) shows the DFE and fig 2.12(b) shows the 2 sample FDTS/DF channel results.

(a) PR4

(b) EPR4

Figure 2.11: PR4 and EPR4 Calculated and Simulated required channel SNR, electronics noise

(a) DFE

(b) FDTS/DF

Figure 2.12: DFE and FDTS Calculated and Simulated required channel SNR, electronics noise

The all cases, there is a good correlation between the calculated channel SNR and the simulated channel SNR. The calculated values are slightly less than the simulated values. However, the performance as a function of density is very consistent showing that the analysis of the detectors is describing the major detector error mechanisms.

2.10.2 Electronics Noise and Media Noise

Fig 2.13(a) shows the calculated and simulated required channel SNR for the PR4 channel to achieve a bit error rate of $10^{-6}$ over a range of recording densities from 1.5 to
3.5 bits per $P_{W_{50}}$ with an equal amount of electronics and medium noise. Fig 2.13(b) shows the EPR4 channel results, fig 2.14(a) shows the DFE and fig 2.14(b) shows the 2 sample FDTS/DF channel results.

![Graph](image.png)

(a) PR4  
(b) EPR4

Figure 2.13: PR4 and EPR4 Calculated and Simulated required channel SNR, electronics and medium noise

![Graph](image.png)

(a) DFE  
(b) FDTS/DF

Figure 2.14: DFE and FDTS Calculated and Simulated required channel SNR, electronics and medium noise

Again, there is good agreement between the calculated results and the simulated values.

### 2.11 Comparison of Detectors

Fig 2.15 shows the simulated performance of the various detectors under electronics noise for a bit error rate of $10^{-6}$. The calculated performance of the matched filter bound (MFB) and MLSD are also plotted to provide an upper bound on system performance. At a recording density of 2.0, the DFE and FDTS/DF detectors perform similarly to the PR4
detector with the EPR4 detector about 0.5dB better. At a density of 2.5, the DFE detector is about 0.4dB better than the PR4 detector. The EPR4 detector is 1.1dB better than the PR4 detector and the FDTS/DF is an additional 0.2dB better than EPR4. At a density of 3.0, the DFE, EPR4 and FDTS/DF detectors are all at a similar level and all about 2dB better than the PR4 detector.

![SNR vs Recording Density Graph](image)

**Figure 2.15: Comparison of detectors with electronics noise**

The PR4 performs worst at high recording densities, while around a density of 2.5 the EPR4 and FDTS/DF detector perform significantly better. The DFE and PR4 detectors perform similarly at low densities but the DFE performance does not drop off as fast as the PR4 detector as the density increases. This is due to the fact that the DFE detector has a different target response chosen to best fit the density of operation while the PR4 has a fixed response best suited to moderate recording densities. At present, typical recording densities for hard disk drives are around 2.0 bits per $PW_{50}$ and increasing. The recent presence of a DFE and EPR4 product in the industry indicates the trend towards densities of 2.5 and greater.

The matched filter bound and MLSD detector are equal up to a density of 2.5 bits per $PW_{50}$. This shows that the inherent ISI in the channel does not necessarily reduce the potential performance of the channel below this density. At a density of 2 bits per $PW_{50}$, the EPR4 detector performs within 1dB of MLSD. At high densities, the best detector
(FDTS/DF) performs within 2dB of MLSD.

Fig 2.16 shows the simulated performance of the various detectors under equal electronics noise and medium noise. The relative performance of the detectors is broadly similar to the electronics noise only case. If effect, the addition of medium noise shifts the recording density axis towards higher densities. For example, the EPR4 detector performs 0.5dB better than the PR4 detector at a density of 2.5 bits per $PW_{50}$ in the presence of equal electronics and medium noise. The FDTS/DF detector performs the best in fig 2.16, justifying its potential for future detectors. At a density of 2.5 bits per $PW_{50}$, it is within 1dB of the MLSD detector.

![Figure 2.16: Comparison of detectors with electronics and medium noise](image)

The actual levels of medium noise and electronics noise will depend on the detailed implementations of the channel components. However, from a detection point of view, the electronics noise component is more difficult to deal with as it is at the channel output and is not filtered by the channel response.

### 2.12 Conclusions

In this chapter an analytical model of the magnetic recording channel has been developed. Analysis of the equalization and detectors for four detection systems have been
considered. These include the PR4 partial response and detector which is currently the most widespread sampled data channel. The EPR4 channel and DFE are also likely to be used and have commercial realizations available. The FDTS/DF is included as a detector which is proposed as a future alternative for high density magnetic recording. The performance of MLSD detection with the matched filter has also been calculated to represent the optimum detection method.

A software model has also been developed which allows a time domain simulation of the various detectors.

The results of calculations and simulations are shown to be in good agreement, verifying the analysis of the detection systems.

This model and simulation software provides a platform for the comparison of new detection schemes proposed later in this thesis with the existing detectors.
Chapter 3

Matched Spectral Codes

3.1 Introduction

This chapter outlines the principles of Matched Spectral Null (MSN) codes. In practice these codes can display problems with catastrophic sequences and may require long path memories to decode. The use of a zero disparity time varying trellis is proposed as a method of avoiding this problem. This method is then shown to extended to zero disparity DC free codes on the Dicode channel. Such a code is developed and implemented in chapter 4. The performance of such coding schemes on the magnetic recording channel is considered and the use of MSN coding with the PR1 partial response target is proposed. This is shown to achieve significant gains with the Lorentz channel model.

3.2 Matched Spectral Null Principle

The principle of MSN codes was developed by Karabed and Siegel in [KS91]. They showed that if a code with symbols from an integer alphabet and an $N^{th}$-order null in its power spectral density at a frequency $f$ is passed through certain partial response channels with an $M^{th}$-order null in their frequency response at the frequency $f$, then the output sequences from the partial response channel will have a minimum distance between them of $\sqrt{2(N+M)}$.

The theorem fits in well with the intuitive idea of not sending signal power at frequencies where the channel response has nulls. It also provides a basis to allow coding for partial response channels that take advantage of the memory inherent in the partial response.

3.2.1 Application to Magnetic Recording

While the matched spectral null theorem applies in general, it is of particular interest for use in magnetic recording channels. This is due to the requirement for binary level input signals. This requirement precludes the use of multilevel trellis coded modulation
schemes that have proved of benefit in modern applications [LM94]. The matched spectral null theorem allows for the systematic design of a coding scheme which allows a binary level mapping and promises high rates. Of particular interest in the magnetic recording channel is the class IV partial response or PR4. This is commonly implemented as two Dicode detectors operating at half the channel rate. The Dicode channel response is characterized as \( 1 - D \) and has a frequency response with a first order null at DC.

The Dicode channel has a minimum distance of \( 2\sqrt{2} \) with uncoded data and \( \pm 1 \) channel input symbols. Hence, if a code with a first order null at DC is used on the Dicode channel, the minimum distance for output sequences on the Dicode channel will be increased to 4.

The use of partial response channels, complicates the implementation of coding schemes due to the memory of the partial response channel. Using precoding to nullify the channel memory has been proposed by [WU86], and then known convolutional codes are used to achieve coding gain. Using this method, a number of coding schemes have been proposed. However, matched spectral codes actually use the channel memory and add additional structure to the data to achieve coding gain and appear to yield less complex decoders than the convolutional approach [KS91].

### 3.3 DC Free Codes

In many communications and storage applications, DC free line codes are used. For example, in helical scan magnetic recording the rotating heads are normally coupled to the stationary circuits via a rotating transformer which cannot pass a DC signal. Normally these line codes are used for their spectrum characteristics, however, the matched spectral null theorem allows coding gain to be achieved by using similar codes. In this section, the properties of such sequences are considered. First finite sequences of symbols are considered.

#### 3.3.1 Finite Sequences

Consider the sampled sequence \( \{x_0, \ldots, x_{N-1}\} \) of \( N \) elements from an integer alphabet with an interval of \( T \) between each sample. The discrete time Fourier transform of the sequence is

\[
X(f) = \sum_{k=0}^{N-1} x_k e^{-j2\pi ft}
\]  

The energy in this signal at a frequency \( f \) is \( |X(f)|^2 \). If this signal has a zero energy at DC then \( X(0) = 0 \) and the Running Digital Sum \( RDS_0(x) \) defined as

\[
RDS_0(x) \equiv \sum_{k=0}^{N-1} x_k = 0
\]  

62
Consider two sequences \( \{ \hat{x} \} \) and \( \{ \tilde{x} \} \) as any two different sequences both meeting the \( RDS_0(x) = 0 \) criteria. The Euclidean squared distance \( d^2 \) between the two sequences can be calculated as

\[
d^2 = \sum_{k=0}^{N-1} (\hat{x}_k - \tilde{x}_k)^2 = \sum_{k=0}^{N-1} e_k^2
\]

Both running digital sums \( \sum_{k=0}^{N-1} \hat{x}_k = 0 \) and \( \sum_{k=0}^{N-1} \tilde{x}_k = 0 \) and \( \{ e_k \} \) is the difference between the two sequences. Hence

\[
\sum_{k=0}^{N-1} \hat{x}_k = 0 = \sum_{k=0}^{N-1} (\tilde{x}_k + e_k)
\]

But

\[
\sum_{k=0}^{N-1} \tilde{x}_k = 0
\]

\[
\Rightarrow \sum_{k=0}^{N-1} e_k = 0
\]

Because the two sequences are different, there must be at least one non-zero element in \( \{ e_k \} \), say \( e_p \). As the original symbols are from integer alphabets, \( |e_p| \geq 1 \). However, as \( \sum_{k=1}^{N} e_k = 0 \), there must exist another element \( e_q \) with \( p \neq q \) and \( |e_q| \geq 1 \) also. Hence, the Euclidean squared distance between any two sequences is

\[
d^2 = \sum_{k=0}^{N-1} e_k^2 \geq |e_p|^2 + |e_q|^2 \geq 2
\]

\[
\Rightarrow d_{\text{min}} \geq \sqrt{2}
\]

where \( d_{\text{min}} \) is the minimum Euclidean squared distance between any two sequences. Hence, if a data source generates integer sequences that have zero energy at DC over finite length, then the sequences have a \( d_{\text{min}} \geq \sqrt{2} \).

In the uncoded case the minimum distance between codewords is

\[
d_{\text{min}}^{\text{uncoded}} = \sqrt{(1 - 0)^2} = 1
\]

When maximum likelihood decoding is used to decode a sequence with a minimum distance of \( d_{\text{min}} \) in the presence of white noise with a variance \( \sigma_n^2 \) the symbol error performance is approximated at high SNR by [For72]

\[
P_{\text{error}} \approx K Q\left(\frac{d_{\text{min}}}{2\sigma_n}\right)
\]

with \( K \) a small constant. Thus, by coding, the system can withstand an additional 3\( dB \) noise power for the same error rate performance.

In the case where the transmitted symbols are elements of the alphabet \( \{+1, -1\} \), the uncoded \( d_{\text{min}} \) is 2 and the coded \( d_{\text{min}} \) is \( 2\sqrt{2} \).
3.3.2 Infinite Sequences

Now, take the sequence \( \{x_k\} \) of length \( N \) starting at time 0 and ending at \( NT \) and consider the power spectrum

\[
P(f) = \lim_{N \to \infty} \frac{1}{NT} \sum_{k=0}^{N-1} x_k e^{-j2\pi ft} \tag{3.11}
\]

Define \( RDS_f(x) \) as

\[
RDS_f(x) = \sum_{k=0}^{N-1} x_k e^{-j2\pi fkT} \tag{3.12}
\]

For a null in the power spectrum of an infinite sequence at frequency \( f \), the \( RDS_f(x) \) must be bounded.

\[
B_1 \leq RDS_f(x) \leq B_2 \quad \text{with} \quad B_1 \leq B_2 \tag{3.13}
\]

The diagram in fig 3.1 shows an example trellis representing a sequence with a \( RDS_0(x) \) that is bounded by \(-3 \leq RDS_0(x) \leq 3\). Each element of the sequence is from the bipolar set \{+1, -1\}. Each state of the trellis represents the current running digital sum of the sequence \( RDS_0 \) and the edges are labeled with the output symbols corresponding to a transition from one state to another. Consider any two paths that diverge at a particular

![Trellis Diagram](image)

Figure 3.1: Trellis for \(-3 \leq RDS_0(x) \leq 3\)

state and re-merge at a later (possibly different) state. One of the paths from the start state will have a +1 while the other will have a -1. When the paths merge to the end state, the same applies and the minimum distance between such paths will be lower bounded by

\[
d_{\text{min}} \geq \sqrt{(1 - (-1))^2 + (1 - (-1))^2} = 2\sqrt{2} \tag{3.14}
\]

Thus, an encoded sequence generated by the trellis in fig 3.1 should have a 3dB coding gain. In general, this gain is not achieved because some sequences in the trellis never merge and have a distance between them smaller than \( d_{\text{min}} \). Such sequences are known as catastrophic sequences.
As an example of a catastrophic sequence, consider two paths starting from state 0 in fig 3.1. One path outputs a symbol +1 and goes to state +1 and the other outputs a symbol −1 and goes to state −1. The accumulated squared distance between the two paths is then 4. However, both paths may now support identical sequences such as −1,+1,−1,+1,⋯ and never merge. Thus, the two paths achieve a distance between them of 2.

3.3.3 Sample DC Free Code

In this section, the design of a DC free code to achieve some coding gain is considered. One of the simplest methods of doing this is to map k bits of input data onto n symbols of output data with no memory between codewords. Each codeword has a RDS0 of 0. This is known as a zero disparity code. Assume the output symbols are from the set \{+1,−1\}. If a block of N such symbols are used then the number of these blocks that are DC free is \(\frac{N!}{(N/2)!^2}\) and N must be even [Imm91]. Table 3.1 lists some sample values. When \(N = 8\) there are 70 possible DC free codes and 6 input bits can be readily encoded. If the two code words \{-1,-1,-1,-1,+1,+1,+1,+1\} and \{+1,+1,+1,-1,-1,-1,-1,-1\} are eliminated then \(-3 \leq RDS0 \leq +3\), otherwise \(-4 \leq RDS0 \leq +4\). If it is assumed that the decoder is synchronized to the odd/even RDS values then, the trellis in fig 3.2 could be used to implement a Viterbi decoder for this code. In this trellis each branch corresponds to two channel symbols.

<table>
<thead>
<tr>
<th>(N)</th>
<th>(N!/(\langle N/2 \rangle!)^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>70</td>
</tr>
<tr>
<td>10</td>
<td>252</td>
</tr>
<tr>
<td>12</td>
<td>942</td>
</tr>
<tr>
<td>14</td>
<td>3432</td>
</tr>
<tr>
<td>16</td>
<td>12870</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>184756</td>
</tr>
</tbody>
</table>

Table 3.1: Number of Zero Disparity Codewords of Length \(N\).

When such a detector is implemented it can produce data errors in the decoded data even with no noise on the received signal. The reason for this comes from the effect of non-converging paths. To see why this happens, consider the trellis in fig 3.2. In a typical Viterbi detector, the data released would be from a particular path with the assumption that all the paths had merged. In this example the released data will be from the survivor path of state 0. Note that a received sequence of \{-1,+1\} or \{+1,-1\} causes each state
to extend its survivor with its previous survivor and hence, no path converging takes place. Table 3.2 shows an example of a sequence that causes the survivor path for state \( \theta \) to be incorrect. In the survivor paths a 0 corresponds to a \(-1\) symbol. The * paths indicate decisions made where the comparison between metrics was comparing equal numbers. In these cases the most unfavorable outcome was selected for demonstration purposes, but a similar result could be expected due to numerical noise or a very small level of noise on the signal. The correct survivor sequence should be \(< 001111>\), but the survivor for path \( \theta \) is incorrect at this point. If this occurred at the end of the path memory then incorrect bits would be released. The length of the path memories does not eliminate the problem as any number of \{-1, +1\} or \{+1, -1\} sequences can be received without changing the metrics or causing paths to converge. Increasing the path memory length does reduce the probability of such sequences occurring. In a simulated example with pseudo random input data and no noise, a bit error rate of \( \approx 1 \times 10^{-5} \) was obtained with a 32 bit path memory, and \( \approx 2 \times 10^{-5} \) with a 64 bit path memory.

<table>
<thead>
<tr>
<th>Input</th>
<th>( \Gamma_{-2} )</th>
<th>Path -2</th>
<th>( \Gamma_{0} )</th>
<th>Path 0</th>
<th>( \Gamma_{+2} )</th>
<th>Path +2</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1, -1</td>
<td>0</td>
<td>&lt; &gt;</td>
<td>0</td>
<td>&lt; &gt;</td>
<td>0</td>
<td>&lt; &gt;</td>
</tr>
<tr>
<td>+1, +1</td>
<td>0</td>
<td>&lt; 0011 &gt;</td>
<td>0</td>
<td>&lt; 001 &gt;</td>
<td>0</td>
<td>&lt; 0010 &gt;</td>
</tr>
<tr>
<td>+1, +1</td>
<td>0</td>
<td>&lt; 001111&gt;</td>
<td>-4</td>
<td>&lt; 0010111 &gt;</td>
<td>-4</td>
<td>&lt; 00101011 &gt;</td>
</tr>
</tbody>
</table>

Table 3.2: Operation of Viterbi Detector on \(-3 \leq RDS_0 \leq +3\) Trellis.

One solution to prevent this undesirable situation occurring is to release the data based on the survivor path that has the maximum metric. This requires additional computations but prevents the possibility of errors in the absence of noise. The simulated performance of such a decoder is shown in fig 3.3 with path memories of lengths of 32 and 64 bits. The
coding gain achieved by this decoder was \( \approx 2dB \) with random data. This is considerable lower than the expected \( 3dB \) gain. The reason for the lower performance is that there exists paths that do not merge and accumulate less than the minimum distance.

![Graph](image)

Figure 3.3: Simulated performance of decoder selecting maximum survivor path.

### 3.4 Time Varying Trellis Structure

An approach has been identified to avoid these problems. It is based on the assumption that the sequence has been synchronized and the start of each 8 symbol codeword is known. This implies that a time varying trellis, as shown in fig 3.4, can be used. At the end of each block of 8 symbols, the path corresponding to the 0 path must be the only survivor. The 8 data bits can then be released and the metrics reset. This brings the advantages of a limited metric range and a small path memory. The time varying trellis can be directly implemented in a programmed flow easily and only calculating the required metrics at each time. The following algorithm demonstrates the principle.

Start: Receive symbol 1.

\[
\Gamma_{+1}^1 = \lambda_{+1} \\
\Gamma_{-1}^1 = \lambda_{-1}
\]

Receive symbol 2.

\[
\Gamma_{+1}^2 = \Gamma_{+1}^1 + \lambda_{+1} \\
\Gamma_{-1}^2 = \max(\Gamma_{+1}^1 + \lambda_{-1}, \Gamma_{-1}^1 + \lambda_{+1}) \\
\Gamma_{-2}^2 = \Gamma_{+1}^1 + \lambda_{-1}
\]

Receive symbol 3.
Receive symbol 8.
\[ \Gamma_0^8 = \max(\Gamma_{+1}^7 + \lambda_{-1}, \Gamma_{-1}^7 + \lambda_{+1}) \]
Release 8 bit path of data.
GOTO start

For a full hardware implementation, the Viterbi algorithm can be implemented with the two sample per step trellis in fig 3.2. However, at the end of each synchronized codeword, the path metric for state \( \theta \) is set to 0 and all the others are set to \(-\infty\). The path memories can be limited to eight bits. The simulated performance of such a system is shown in fig 3.5, and achieves in excess of 3\( dB \) coding gain with random data.

![Trellis Diagram](image)

**Figure 3.4:** Time varying trellis, rate 6/8 code.

3.4.1 Run Length Constraints

As the DC free codes are normally used for line coding, their run length properties are also of concern. However, the requirement for a bounded RDS also limits the run length of identical symbols. Consider a DC free sequence whose RDS has achieved its lower bound of \( B_1 \). If the sequence then has a run of +1 symbols, there can be at most \( B_2 - B_1 \) consecutive such symbols before the RDS reaches the upper bound. Hence, the run length of identical symbols is at most \( B_2 - B_1 \). For the trellis of fig 3.4, the RDS is bounded with \(-3 \leq \text{RDS} \leq +3\) and the maximum run length of identical symbols is 6.

3.5 Higher Order Nulls

DC free codes with higher order spectral nulls may also be constructed and should achieve a higher coding gain than codes with a 1st-order null. In this section, such codes

---

1 Or the most negative value possible in the implementation
Figure 3.5: Simulated BER performance of time varying trellis, rate 6/8 code.

are considered and an example 2nd-order null code using a time varying trellis is described.

Previously the spectrum of a sequence \( \{x_1, x_2, \ldots, x_N\} \) was written as

\[
X(f) = \frac{1}{NT} \sum_{n=1}^{N} x_n e^{-j2\pi fnT}
\]  

The power spectrum of the sequence can be written as

\[
P(f) = |X(f)|^2 = \frac{1}{NT} \sum_{n=1}^{N} e^{-j2\pi fnT} \frac{1}{NT} \sum_{n=1}^{N} e^{j2\pi fnT}
\]

A higher order null is defined as a frequency where the power spectrum becomes zero and some of its higher order derivatives also become zero. To evaluate these, define

\[
F(a, b, f) \equiv \sum_n n^a x_n e^{-j2\pi fnT} \sum_m m^b x_m e^{j2\pi fmT}
\]

and

\[
RDS_k^f(x) \equiv \frac{n^a x_n e^{-j2\pi fnT}}{n}
\]

\[
\Rightarrow F(a, b, f) = RDS_k^f(x)(RDS_k^f(x))^* 
\]

\[
F(a, b, f) = 0 \begin{cases} RDS_k^f(x) = 0 \\ RDS_k^f(x) = 0 \end{cases}
\]

To calculate the higher order derivatives of the power spectrum of the sequence, the following can be used

\[
\frac{d}{df}(F(a, b, f)) = j2\pi T(F(a, b + 1, f) - F(a + 1, b, f))
\]
The power spectrum of the sequence can be written as

\[ P(f) = \frac{1}{(NT)^2}F(0,0,f) \] (3.22)

This is 0 if \( RDS_0^f x = 0 \) as before. The derivatives of \( P(f) \) are

\[
\frac{d}{df} P(f) &= \frac{j2\pi T}{(NT)^2} (F(0,1,f) - F(1,0,f)) \\
\frac{d^2}{df^2} P(f) &= \frac{(j2\pi T)^2}{(NT)^2} (F(0,2,f) - 2F(1,1,f) + F(2,0,f)) \\
\frac{d^3}{df^3} P(f) &= \frac{(j2\pi T)^3}{(NT)^2} (F(0,3,f) - 3F(1,2,f) + 3F(2,1,f) - F(3,0,f)) \\
\frac{d^4}{df^4} P(f) &= \frac{(j2\pi T)^4}{(NT)^2} (F(0,4,f) - 4F(1,3,f) + 6F(2,2,f) - 4F(3,1,f) + F(4,0,f)) \\
&\vdots
\] (3.23)

The first derivative of the power spectrum at a frequency \( f \) is zero if there is a null at that frequency. However, for the second derivative to be zero at a frequency \( f \), the value of \( RDS_1^f (x) = 0 \). This implies the third order derivative is zero. The fourth order derivative is zero if \( RDS_2^f (x) = 0 \). In general, the \( 2k^{th} \) derivative is zero if and only if \( RDS_k^f (x) = 0 \).

A sequence is said to have an order-K spectral null at the frequency \( f \) if its derivatives of order 0 to \( 2K-1 \) are zero at the frequency \( f \). Hence, a order-K spectral null at \( f \) implies that

\[ RDS_k^f (x) = 0 \quad \text{for} \quad k = 0, \ldots, K-1 \] (3.24)

\[ \Rightarrow \sum_n n^k x_n e^{-j2\pi fn} = 0 \quad \text{for} \quad k = 0, \ldots, K-1 \] (3.25)

It is shown in [Imm91] that the minimum distance for such sequences is \( d_{\text{min}}^k \geq 2(K) \). This implies higher coding gains for higher order DC nulls.

### 3.5.1 Example Higher Order DC Null Code

<table>
<thead>
<tr>
<th>2nd-Order Null Codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>{-1, -1, +1, +1, +1, +1, -1, -1}</td>
</tr>
<tr>
<td>{-1, +1, -1, +1, +1, -1, +1, -1}</td>
</tr>
<tr>
<td>{-1, +1, +1, -1, -1, +1, +1, -1}</td>
</tr>
<tr>
<td>{-1, +1, +1, -1, +1, -1, -1, +1}</td>
</tr>
<tr>
<td>{+1, -1, -1, +1, -1, +1, +1, -1}</td>
</tr>
<tr>
<td>{+1, -1, +1, -1, +1, +1, -1, +1}</td>
</tr>
<tr>
<td>{+1, -1, -1, -1, -1, -1, -1, -1}</td>
</tr>
</tbody>
</table>

Table 3.3: 2\textsuperscript{nd}-Order DC null codewords of length 8.
Consider the list of codewords in table 3.3. They each meet the requirement that 
\[ \sum_{n=1}^{8} n x_n = 0 \] and \[ \sum_{n=1}^{8} x_n = 0 \] and hence, should have a 2\textsuperscript{nd}-order spectral null at DC. This implies a free distance of \( d_{\text{min}} \geq 4 \) with \pm 1 symbols. The trellis in fig 3.6 could be used to soft decode this code. Each node is labeled with the \( RDS_0(x)/RDS_1(x) \). The results of simulating this code is shown in fig 3.7 and shows the expected 6dB coding gain.

![Decoding trellis for 2\textsuperscript{nd}-order DC null code](image)

**Figure 3.6: Decoding trellis for 2\textsuperscript{nd}-order DC null code**

![BER performance of rate 3/8 code with 2\textsuperscript{nd}-order null](image)

**Figure 3.7: BER performance of rate 3/8 code with 2\textsuperscript{nd}-order null**

### 3.6 Partial Response Channels

It has been demonstrated that DC sequences have desirable distance properties, however, to obtain the potential gain from such codes requires avoiding problems with catastrophic sequences. It has been shown that the use of a time varying trellis and decoder can eliminate catastrophic sequences and achieve the desired coding gain. In this section, it is
illustrated how codes with a spectral null are used to achieve coding gain with partial response channels. A design describing the full implementation details of a rate 6/8 code for a PR4 channel is presented in chapter 4.

3.6.1 The Dicode Channel

The most common partial response channel in magnetic recording is the PR4 channel with a partial response polynomial of $1 - D^2$. This can be used as two completely independent $1 - D$ channels by considering all odd bits to be from one $1 - D$ channel and all even bits from the other $1 - D$ channel. This is readily done by deinterleaving. Fig 3.8 shows the trellis for the $1 - D$ channel.

![Trellis for $1 - D$ dicode channel](image)

These Dicode channels have the property of a null in their frequency response at DC. From the principle of matched spectral codes, it should be possible to achieve coding gain by using a code with a DC null. The output sequences will have a 2nd-order null at DC and hence, a minimum distance of $d_{min} \geq 4$.

3.6.2 Combined Trellis

Fig 3.9 shows the trellis for a running digital sum constraint of $-3 \leq RDS_0(x) \leq 3$. If this trellis and the Dicode trellis of fig 3.8 are concatenated then a suitable trellis for decoding the output of the Dicode channel with a DC free code can be obtained.

The concatenated trellis can have all combinations of the states in both the Dicode channel trellis and the bounded RDS trellis. If the states in the concatenated trellis are labeled (rds value/dicode state) then the desired trellis is shown in fig 3.10. This trellis has a minimum free distance of 4 with the outputs labeled $\{-2, 0, +2\}$. The trellis repeats every two samples and can also be drawn as in fig 3.11. In this case two input samples are used for each branch. This would allow faster throughput at the expense of requiring a more complex implementation.

72
3.6.3 Catastrophic Sequences

As in the case of DC free codes there are infinite length sequences that never merge and do not gain the minimum distance. Consider the trellis in fig 3.10 with two paths emanating from the state 0/+1. One path goes to state +2/+1 with the outputs {0, 0} and continues indefinitely with the repeating pattern {−2, +2, −2, +2, . . .}. The other path goes to state 0/+1 with the outputs {−2, +2} and then continues indefinitely with the repeating pattern {−2, +2, −2, +2, . . .}. The distance achieved between these two paths is \( \sqrt{(-2)^2 + 2^2} = 2\sqrt{2} \) and they continue indefinitely without merging or gaining any increased distance. Hence, any detector with a finite length path memory will not be able to resolve this path with more than a distance of \( 2\sqrt{2} \). This is the same minimum distance
as an uncoded PR4 channel.

### 3.7 Time Varying Trellis

However, if a time varying trellis is used as previously described for the case of DC free codes, then a combined time varying trellis may be constructed. With a zero disparity code of length $L$, the RDS of the sequence becomes 0 every $L$ symbols. When this code trellis is combined with the Dicode trellis then the resulting time varying trellis as shown in fig 3.12 results. It will now be shown that this time varying trellis guarantees to accumulate a sufficient distance between unmerged paths to eliminate catastrophic sequences.

#### 3.7.1 Accumulated Distance Between Unmerged Paths

At the end of each codeword, there are only two possible states $0/-1$ and $0/+1$. From the definition of each state and the channel function of $1 - D$, a number of properties for the trellis can be derived.

Consider a state with a running digital sum RDS and Dicode state $-1$ i.e. $\text{RDS}/ -1$,

1. If the input bit is a logic 0 then the channel output symbol is 0 and the next state is $\text{RDS}-1/ -1$.

2. If input bit is a logic 1 then the channel output symbol is $+2$ and the next state is $\text{RDS}+1/ +1$.

Consider the state $\text{RDS}/ +1$,
3 If the input bit is a logic 1 then the channel output symbol is 0 and the next state is \( \text{RDS} +1 / +1 \).

4 If input bit is a logic 0 then the channel output symbol is \(-2\) and the next state is \( \text{RDS} -1 / -1 \).

For the state \( 0/-1 \), the only possible outputs from this state is \(+2\) or \(0\). The only possible outputs from the state \( 0/+1 \) are \(-2\) and \(0\). Consider two paths, one starting from the state \( 0/-1 \) and the other starting from the state \( 0/+1 \) at the beginning of a codeword. If the two paths do not gain any distance between them, then their channel outputs must be a pattern of \( \{0, 0, \ldots, 0, 0\} \). In this case the path \( 0/-1 \) goes to \( -1/-1 \) and the RDS value keeps decreasing. The state \( 0/+1 \) goes to \( +1/+1 \) and the RDS value keeps increasing. After \( N \) 0 outputs the two paths would be in the states \( -N/-1 \) and \( N/+1 \). Hence, they cannot return to a RDS value of \( 0 \) which is required for the trellis to converge at the end of the codeword. Therefore, any paths that converge to the allowable end states must have at least one symbol which is different.

Take the case of one of the paths having a different output at some point. This may occur in a number of cases.

- Both paths converge. In this case the error event must have accumulated a minimum distance of 4 as guaranteed by the matched spectral null theorem.

- One path has an output of \( -2 \) and the other has an output of \(+2\). In this case the paths accumulate a squared distance of 16 which is the expected squared minimum distance of the channel output.
- One path has an output of −2 and the other has an output of 0. In this case the paths would have been in the states \(-N/-1\) and \(N/+1\) and are now in the states \(-N-1/-1\) and \(N-1/-1\). The accumulated squared distance will be increased by 4 but they have different RDS values and the same last symbol of −1. At the end of the trellis they must converge to the same RDS value of 0. If the paths are to remain unconverted at the end of the codeword, then they must end in states \(0/-1\) and \(0/+1\). This requires that the paths have an additional difference in the channels outputs, thus increasing the accumulated squared distance to at least 8.

Hence, any two paths starting in the states \(0/-1\) and \(0/+1\) and ending in those states after \(L\) channel symbols will either have merged or will have accumulated a squared distance of at least 8.

Now, consider error events starting in a codeword. Take an arbitrary state \(N/-1\) and consider two paths diverging from this state. One path will have an input bit of a logic 0 and output symbol of 0 and will go to state \(N-1/-1\). The other path will have an input bit of a logic 1 with an output symbol of +2 and will go to state \(N+1/+1\). In this case the accumulated square distance between the paths is 4, and the paths are ending in the states \(N-1/-1\) and \(N+1/+1\).

Only channel symbols +2 and 0 may come from the state \(N-1/-1\) and only symbols −2 and 0 may come from the state \(N+1/+1\). Hence, no additional distance will be accumulated between the paths only if the output symbols are a string of 0’s from each state. In this case the state \(N-1/-1\) will follow the pattern

\[N-1/-1, N-2/-1, \ldots, N-M/-1\]

and the state \(N+1/+1\) will follow the pattern

\[N+1/+1, N+2/+1, \ldots, N+M/+1\]

Thus, the RDS values will diverge and cannot converge to the states \(0/-1\) and \(0/+1\) required at the end of the trellis.

Therefore, in order for the two paths to possibly converge to the allowed states at the end of the codeword, there must be an additional difference in the channel outputs from both paths. The accumulated squared distance will then be 8 at the end of the codeword.

These results may be restated as:

1. Any two distinct paths at the start of a codeword will accumulate an additional squared distance of 8 if they have not merged by the end of the codeword.

2. Any two distinct paths starting within a codeword will accumulate a squared distance of 8 if they have not merged by the end of the codeword.
These results imply that if a path memory of depth $2L$ symbols is used with a zero disparity code of length $L$ and time varying trellis, then any unmerged paths of this length will have accumulated a squared distance of at least 16 and one can be discarded without a loss in coding gain.

This allows for the design of a time varying MSN code for use on the dicode channel. In chapter 4, a rate 6/8 code for use on the PR4 channel is developed. A time varying detector is described and a VLSI implementation presented.

### 3.7.2 Non-Catastrophic Codes

It is possible to design codes that do not have any catastrophic sequences and can be used without a time varying trellis. This can be done by observing that a sequence of data that hits both the upper and lower bounds of the allowable RDS values will have no catastrophic sequences. Fig 3.7.2 shows an example trellis with a highlighted path meeting both bounds. Dotted lines indicate potential catastrophic sequences and these can be seen to terminate as the actual path meets the boundary.

![Figure 3.13: Example path meeting both bounds](image)

This is how the codes presented in [KS91] were designed but this results in very long path memory requirements and error events. The rate 8/10 code described in [TRS+92] required a path memory of 64 bits.

### 3.7.3 Other Time Varying Structures

The principle of a time varying trellis has also been recently presented in the literature. The patents [Fre94] and [Fre93] describe similar methods that eliminate catastrophic sequence through time varying structures. These methods and variations are described in [FKR+95], and used in [RCS+95] to design a rate 8/10 MSN code and detector with a time varying trellis.
3.8 Performance of MSN Codes on the Recording Channel

While MSN codes provide a significant coding gain, they suffer a rate loss compared to normally used line codes. To date, the highest rate MSN codes to be implemented are of rate 8/10, and higher rates are likely to require considerably more complex encoder/decoder logic and more states in the Viterbi decoder. Most existing PRML channels use rate 8/9 codes.

MSN codes can be employed in a number of ways. The track density can be increased until the same error rate as in the uncoded case is achieved and an increase in recording density achieved.

Consider the case of an equal contribution of electronics noise $N_e$ and medium noise $N_m$ on a given channel. If the track width was reduced by a factor $\alpha$, the signal power $S$ would reduce to $\alpha^2 S$ and the medium noise would reduce to $\alpha N_m$, assuming it is uncorrelated across the width of the track. The electronics noise would remain unchanged. The original

\[
SNR_0 = \frac{S}{N_e + N_m}
\]

while the new SNR is

\[
SNR_1 = \frac{\alpha^2 S}{N_e + \alpha N_m}
\]

The additional noise power allowed due to the use of the MSN code is 3dB and with equal media and electronics noise $N_e = N_m$, allowing $\alpha$ to be calculated from

\[
\frac{S}{N_e + N_m} = 2 \frac{\alpha^2 S}{N_e + \alpha N_m} \quad \alpha > 0
\]

from which $\alpha = 0.64$ or a track density increase of 1.56. Thus, for a code rate of 6/8 the resulting increase in recording density is $1.56 \times \frac{6}{8}/\frac{8}{9}$ or 31%, and with a rate 8/10 code, the resulting increase in recording density is 40%. However, track densities are limited by mechanical factors and the positional accuracy of servos, and may not be easily able to accommodate a 56% increase.

It is more likely that the coding gain from the MSN code would be used to increase the linear recording density or yield additional SNR at the same user recording density to increase margins. To achieve the same user recording density with a rate 8/10 encoded MSN code requires increasing the recording density of the channel by a factor of 10/9 or 11% with the associated loss in SNR.

Fig 3.14 shows the simulated performance with electronics noise of the 6/8 time varying trellis MSN code on the PR4 channel and the normal rate 8/9 PR4 channel. The 8/10 code is plotted as the rate 6/8 simulated data with the higher code rate assumed.

From the simulated results, it can be seen that the rate 6/8 MSN coded system performs better than the 8/9 system up to densities of about 2.3 bits per $PW_{50}$. A rate 8/10 MSN
code would perform better with 1.4dB gain at a recording density of 2.0 bits per $PW_{50}$. In [RCS+95], a rate 8/10 MSN implementation is presented and experimental results on a disk drive channel confirms the performance gains of such codes.

3.9 Matched Spectral Codes on the PR1 channel

In general, the previously described principles apply to nulls at any frequency. For example, the EPR4 channel has a $2^{nd}$-order null at the Nyquist frequency and has a minimum distance of 2 with outputs labeled $\{-2, -1, 0, +1, +2\}$. If this was used with a code having a $1^{st}$-order null at the Nyquist frequency, the outputs from the channel should have a minimum distance of $\sqrt{6}$. This is an additional 1.78dB coding gain.

The problems of catastrophic sequences will still be present and have to be avoided. This will always be the case as the matched spectral codes only guarantee minimum distance bounds for merged sequences.

A null at the Nyquist frequency can readily be obtained by inverting every other bit of a sequence with a Null at DC. Nulls at other frequencies are also possible [Imm91].

Initial simulations indicated that the PR1 or $1 + D$ partial response performed better than PR4 at higher densities and thus might be suitable for use with a matched spectral null code. The PR1 response has a null at the Nyquist frequency but no null at DC and hence requires a DC free code for use on the magnetic recording channel. For the PR1 channel, the MSN code required is a code with both a DC null for the magnetic recording channel and a Nyquist Null to match the PR1 response. A code has a DC null if and only
if the Running Digital Sum (RDS) defined as

\[ RDS \equiv \sum_{k=-\infty}^{\infty} x_k \]  \hspace{1cm} (3.29)

is bounded. Similarly, for a null at the Nyquist frequency, the Alternating Running Digital Sum (ARDS) defined as

\[ ARDS \equiv \sum_{k=-\infty}^{\infty} x_k(-1)^k \]  \hspace{1cm} (3.30)

must be bounded.

In order to avoid the problem of catastrophic sequences an interleaved zero disparity code is used. A zero disparity code has its RDS equal to zero at the end of each block. This additional information allows elimination of catastrophic sequences through use of a time varying Viterbi detector as described in section 3.7.

The code used here is based on a rate 6/8 DC free code that uses 64 of the 68 sequences of length 8 which have zero disparity and whose RDS is bounded by \(-3 \leq RDS \leq +3\). This rate 6/8 code has a null at DC. Consider interleaving two such codes to create blocks of 16 symbols. The RDS of each block is

\[
RDS = \sum_{k=0}^{15} x_k \\
= \sum_{k=0,2,\ldots,14} x_k + \sum_{k=1,3,\ldots,15} x_k \\
= 0 + 0 = 0 
\]  \hspace{1cm} (3.31)

The ARDS of each block is

\[
ARDS = \sum_{k=0}^{15} x_k(-1)^k \\
= \sum_{k=0,2,\ldots,14} x_k - \sum_{k=1,3,\ldots,15} x_k \\
= 0 - 0 = 0 
\]  \hspace{1cm} (3.32)

Hence, the interleaved code has a RDS and ARDS of 0 and thus has a null at DC and at the Nyquist frequency as desired. The code has the properties that \(-6 \leq RDS \leq +6\) and \(-6 \leq ARDS \leq +6\). This can be viewed as a rate 12/16 code that can be encoded and decoded with a complexity similar to a rate 6/8 code. The power spectral density of the code can be expressed as an interleaved version of eqn 2.20

\[
S_z(f) = \frac{1}{T} \frac{n}{n-1} \left( 1 - \left( \frac{\sin(n\pi f T)}{n \sin(\pi f T)} \right)^2 \right) 
\]  \hspace{1cm} (3.33)

A Viterbi detector operating on the ARDS and the channel state can be used to detect the channel outputs. The PRI channel has two possible states and the code has 13 possible ARDS values. However, the ARDS of the symbols alternates between odd and even values.
and at each time step only 6 (odd) or 7 (even) ARDS values are possible. The Viterbi detector can be implemented with the equivalent of 12 Add Compare Select (ACS) units.

3.9.1 Probability of Error for PR1 Detector

The probability of error in the presence of correlated noise for the PR1 detector may be calculated as described in section 2.7.3.

![PR1 trellis](image)

Figure 3.15: PR1 trellis

The trellis for the PR1 channel is shown in fig 3.15. Consider all the error events of length 2 starting from state A. These are shown in fig 3.16. A similar diagram may be drawn for error events starting in state B. Counting both, there are 8 possible output sequences each of which can support one minimum distance error event and each minimum distance event causes one bit error. The error event sequence $e_k$ in all cases is $\pm\{+2, +2\}$. This is summarized in table 3.4.

![Error events of length 2 starting in state A](image)

Figure 3.16: Error events of length 2 starting in state A

<table>
<thead>
<tr>
<th>Length</th>
<th>P(event supported)</th>
<th>Number of Bit Errors</th>
<th>$e_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$\frac{8}{5}$</td>
<td>1</td>
<td>$\pm{+2, +2}$</td>
</tr>
</tbody>
</table>

Table 3.4: Minimum distance error events of length 2

Similarly for events of length 3, table 3.5 may be constructed. In general for events of length $l$, table 3.6 may be constructed.

With the error events characterized in this manner, an expression for the probability of
an error event for the PR1 Viterbi detector may be calculated as in section 2.7.3

\[
P(\text{Error Event}) = Q \left( \frac{\sum_{k=0}^{l} \epsilon_k^2 R_n(0) + \sum_{i=0}^{l-1} \sum_{j=0, j \neq i}^{l-1} \epsilon_i \epsilon_j R_n(i-j)}{\sqrt{\sum_{k=0}^{l} \epsilon_k^2 R_n(0) + \sum_{i=0}^{l-1} \sum_{j=0, j \neq i}^{l-1} \epsilon_i \epsilon_j R_n(i-j)}} \right) \tag{3.34}
\]

Using the general form of an error event of length in table 3.6, the bit error probability may be calculated as

\[
P_e(PR1) = \frac{\sum_{i=1}^{\infty} 2i}{2^l} Q \left( \frac{\sqrt{2}}{\sqrt{R_n(0) - (-1)^i R_n(i)}} \right) \tag{3.35}
\]

### 3.9.2 Model Results

Based on the developed model, the performance of the PR4 and EPR4 channels with random data and the PR1 channel with a DC free zero disparity rate 6/8 code and DC and Nyquist free zero disparity rate 12/16 codes have been calculated. The channel is assumed to have 50% medium noise and 50% read electronics noise. Fig 3.17 shows the SNR loss to obtain a bit error rate of $10^{-5}$ as a function of recording density. The SNR is normalized to 0dB for a PR4 channel at a density of 2.0.

It can be seen that as the recording density increases, the PR1 channel performs better than the PR4 or EPR4 and hence is better suited to the use of matched spectral coding. If a matched spectral null code is applied to the PR1 channel with a rate 12/16 code then an expected 3dB coding gain should be obtained. This channel and code requires an additional 3dB SNR at a recording density of 3.25. Hence, the use of a matched spectral null code should allow a recording density of 3.25 to achieve the same error rate as a PR4 channel at a recording density of 2.0. This indicates a potential increase in user data density of $3.25 \times \frac{12}{16}/2.0 \times \frac{8}{9}$ or 37% assuming the use of a rate 8/9 code with the PR4 channel.
3.9.3 Simulated Channel Model

Using the computer model outlined in section 2.9, the following detections schemes were simulated.

- **PR4**: The source encoder is a rate 8/9 (0,4/4) encoder and the target response is \(1 - D^2\). The detector is a 4-state Viterbi Detector with 32-bit path memory.

- **EPR4**: The source encoder is a rate 8/9 (0,4/4) encoder and the target response is \(1 + D - D^2 - D^3\). The detector is an 8-state Viterbi Detector with 32-bit path memory.

- **PR1**: The source encoder must provide sequences that are DC free. A number of possibilities exist such a rate 6/8 zero disparity or rate 8/10 low disparity code[FKSO86]. A 2-state Viterbi Detector with 32-bit path memory is used for the \(1 + D\) channel with a DC free code.

- **PR1+MSN**: The source encoder is a rate 12/16 DC free and Nyquist free encoder, the target channel is a \(1 + D\) response and the detector is a varying trellis decoder with the equivalent of 12 states. A 64-bit path memory is used.

3.9.4 Simulation Results

The performance of the various channels have been simulated and the results are shown in fig 3.18. The SNR loss in obtaining a bit error rate of \(10^{-5}\) as a function of recording density is plotted. The SNR is normalized to 0dB for a PR4 channel at a density of 2.0.
Figure 3.18: Simulated detection performance for various detection schemes versus recording density.

The simulated data is good agreement with calculated results. The plot also shows the simulated performance of the MSN decoded data. The resulting SNRs are similar to the expected values. The results indicate that the PR4 channel at a density of 2.0 performs at the same level as the PR1 channel with MSN coding at a density of 3.1. This is less than the calculated values but shows an increased user data density of $3.1 \times \frac{12}{16} / 2.0 \times \frac{5}{9}$ or 30%.

3.9.5 More Efficient Codes

While the rate 12/16 code performed as required, a 12-state Viterbi detector is quite complex. However, the code for use on the PR1 channel does not require a zero disparity constraint for the DC null. This is only required for the Nyquist null to allow MSN detection. This allows the possibility of using a low disparity DC null [Imm91] [FKSO86].

Consider all the binary codewords of length 12. By an exhaustive search, 328 have a RDS of 0 and ARDS of 0 with $-3 \leq \text{ARDS} \leq +3$ at any point in the codeword, 195 have a RDS of $-4$ and ARDS of 0 with $-3 \leq \text{ARDS} \leq +3$ and 195 have a RDS of +4 and ARDS of 0 with $-3 \leq \text{ARDS} \leq +3$.

A low RDS disparity code can then be implemented to encode 512 9 bit words. Of the possible 512 input words, 328 can be encoded with the codewords that have an RDS of 0. The other 184 input words can be mapped alternately between the codewords with a RDS of +4 and those with a RDS of −4. In this way the RDS at the end of any sequence of codewords is $-4 \leq \text{RDS} \leq +4$. This bounded RDS results in the required null at DC. The ARDS of any codeword is 0 at the codeword boundary and is bounded between $-3$ and +3 at all times. This allows the use of a time varying Viterbi detector to achieve a 3dB
coding gain. The Viterbi detector required to track the ARDS variation of this code only requires 6 states and the code rate is 9/12.

3.9.6 Application of MSN Coding on the PR1 Channel to Tape Drives

MSN coding on the PR1 channel is most advantageous at high recording densities. Future disk drive channels may operate at such high densities but tape drive products are already using high density recording. A good example is the DDS R-DAT helical scan tape drives. The current proposal for the DDS3 standard is considering the use of the PR1 partial response detection system with a rate 8/10 DC free code [FHOH95]. Such a system is also described in [ISH95].

These systems can operate at high recording densities because they use contact recording and suffer less from non-linearities. The channels also operate at higher bit error rates because they use powerful Reed Solomon ECC codes across multiple data blocks. DC free codes are used due to the rotary transformer required for the rotating heads in helical scan recording.

For these reasons, the use of the MSN codes proposed here appear attractive. The MSN codes would function as a short soft decision code. It is known that the use of a short maximum likelihood detected inner code combined with a high rate non-binary outer code provides an efficient method of coding [For96]. The rate loss going from a rate 8/10 code to a rate 9/12 code is quite reasonable in return for a 3dB coding gain.

Fig 3.19 shows the performance of the existing DDS 8/10 code with linear equalization and detection, the 8/10 code with PR1 equalization and detection and the proposed 9/12 code with PR1 equalization and MSN detection. The channel has equal medium and electronics noise. The 8/10 code with linear equalization degrades rapidly with increasing density. The PR1 channel performs significantly better as reported in [FHOH95]. The proposed 9/12 code performs even better at high recording densities. Consider operation of the rate 9/12 code at a recording density of 3.5 bits per \( PW_{50} \). For the same user density, the 8/10 code with PR1 requires operation at a recording density of 3.28 bits per \( PW_{50} \). The required channel SNR for the MSN code is 2dB less illustrating its potential in this application.

3.10 Conclusions

In this chapter, the principles of MSN codes were outlined and the problem of catastrophic sequences was illustrated. A time varying trellis was proposed as a means of avoiding the problem and shown to achieve the desired coding gain with DC free codes. The same principle was applied to a 2nd-order null code.
The principle of the time varying trellis was extended to the use of a DC free code on the Dicode channel. This allows the design of MSN codes for the PR4 channel using a time varying trellis with relatively short path memories.

The performance of such MSN codes on the PR4 magnetic recording channel was considered and their use on a PR1 channel was proposed to provide greater gains in linear density. A suitable code for the PR1 channel was described and shown to provide significant gains. The use of MSN codes with the PR1 channel for tape drive applications was also proposed.
Chapter 4

Design and Implementation of MSN VLSI IC

4.1 Introduction

This chapter illustrates the design, evaluation and implementation of a rate 6/8 matched spectral null code. The principles of matched spectral null codes were described in chapter 3. In order to successfully implement such a code, consideration must be given to all aspects of the encoder and decoder. Fig 4.1 shows a block diagram of the intended system. The code used is a zero disparity code that uses 64 of the possible 68 zero disparity bipolar sequences of length 8 with a RDS variation less than or equal to 6. The code is matched to a $1 - D$ channel and is interleaved to operate with a PR4 channel. The channel output is detected with a time varying Viterbi detector. The code achieves a 3dB coding gain over an uncoded PR4 channel.

4.2 Rate 6/8 Code

A zero disparity code was chosen for the following reasons.

**Catastrophic Sequences:** The use of a zero disparity code means that at the end of each codeword, the RDS becomes zero. Viterbi detection of the code is based on using the RDS value as states and hence, the correct state is known at the end of each codeword. When the code is concatenated with a $1 - D$ channel, there are two possible states at the end of each codeword and using this information avoids the problem of catastrophic sequences.

**Reduced Path Memory:** The required path memory is 16 bits for each interleave. This compares favorably to the 64-bit path depth in [TRS+92].
Low RDS: With zero disparity code words of length 8, 68 have $-3 \leq RDS \leq +3$ and this requires 6 states in the Viterbi detector. A larger RDS variation would require further states.

Implementation: Such a straightforward code can be implemented using a small ROM or logic and can be decoded with no look ahead or memory requirements.

Run Length Constraint: The maximum number of consecutive zero output samples after interleaving is 10. This should allow sufficient information for timing recovery and AGC calculations.

Table 4.1 lists a possible assignment for an encoder. This assignment was heuristically chosen. The assignment is 180 degree shift invariant such that if a source word $A \rightarrow X$ then $\overline{A} \rightarrow \overline{X}$. This implies that if the channel polarity is inverted, then the decoded data will also be inverted. Differential precoding can then be used to eliminate the effect of channel polarity.

4.3 Decoder Trellis

The decoder trellis may be derived by concatenating the trellis of the rate 6/8 encoder and the trellis of the $1 - D$ decoder channel. The Dicode trellis was shown previously in fig. 3.8. The rate 6/8 code trellis is shown in fig 4.2. The combined trellis has states that can be labeled $(dicodestate, rdsvalue)$ and is shown in fig 4.3 with the states labeled as in table 4.2.
The decoder trellis is used to design a time varying Viterbi detector. Each state is only valid every second cycle and hence ACS units and path metrics can be reassigned. Table 4.3 shows one possible combination of assignments.

The trellis varies with a period of 8 input samples. From the trellis in fig 4.3 the
sequence of operations can be determined. Table 4.4 shows the sequence of operations for the \( a1 \) state. Similar sequences can be derived for each of the other states.

<table>
<thead>
<tr>
<th>(Input Sample No.) ( \mod 8 )</th>
<th>( \Gamma_{ain} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\textit{No Operation}</td>
</tr>
<tr>
<td>1</td>
<td>( \Gamma_{a1} = \Gamma_{ch} + \lambda_0 )</td>
</tr>
<tr>
<td>2</td>
<td>( \Gamma_{a1} = \Gamma_{a1} + \lambda_2 )</td>
</tr>
<tr>
<td>3</td>
<td>( \Gamma_{a1} = \text{Max}(\Gamma_{ch} + \lambda_0, \Gamma_{a1} + \lambda_2) )</td>
</tr>
<tr>
<td>4</td>
<td>( \Gamma_{a1} = \text{Max}(\Gamma_{bg} + \lambda_0, \Gamma_{a1} + \lambda_2) )</td>
</tr>
<tr>
<td>5</td>
<td>( \Gamma_{a1} = \text{Max}(\Gamma_{ch} + \lambda_0, \Gamma_{a1} + \lambda_2) )</td>
</tr>
<tr>
<td>6</td>
<td>( \Gamma_{a1} = \text{Max}(\Gamma_{bg} + \lambda_0, \Gamma_{a1} + \lambda_2) )</td>
</tr>
<tr>
<td>7</td>
<td>\textit{No Operation}</td>
</tr>
</tbody>
</table>

Table 4.4: Required operations for the \( a1 \) state

4.4 \textbf{Practical Aspects of the Detector}

4.4.1 ADC Resolution Requirements

For VLSI implementation the precision requirements for calculations need to be determined. Excessive precision results in increased area and power consumption and reduced speed of operation while insufficient precision can result in loss of coding gain. The time varying trellis described has been simulated with floating point precision calculations and a
variable resolution quantizer. The plot in fig 4.4 shows the results. In this case the input is clipped in the range \([-2.0, \ldots, 2.0]\) and quantized to the specified resolution. From the graph, quantization levels of 4 bits and higher produce results very close to the unquantized case. This implies that a 4-bit or higher quantizer is sufficient to achieve the required coding gain. In practical PR4 channels some additional range outside the \([-2.0, \ldots, 2.0]\) is required for timing recovery and AGC calculations.

![Figure 4.4: Rate 6/8 code performance with different quantization levels](image)

### 4.4.2 Finite Precision Arithmetic

The finite precision implementation details of the ACS units also need consideration. This was done based on a 6-bit quantizer. The branch metrics for the expected input values of \(\hat{y} \in \{-2.0, 0.0, +2.0\}\) are in general \(-\left(y_{in} - \hat{y}\right)^2\). Eliminating constant terms and factors yields branch metrics of:

\[
\begin{align*}
\lambda_{-2} &= -y_{in} \\
\lambda_0 &= 1.0 \\
\lambda_{+2} &= +y_{in}
\end{align*}
\]

(4.1)

In this case the new path metrics may be calculated by adding one of the above to the existing path metrics. Since the input \(y_{in}\) is a 6-bit number, the branch metrics above are also representable with a 6-bit number. The range of the branch metrics is from \(-32\) to \(+31\) decimal. By exhaustively checking the trellis of fig 4.3 with all combinations of the input
values +31, −32 and 0, the maximum difference between the maximum value of any metric and the minimum value at each state was 121 decimal. Hence, the maximum difference between path metrics is upper bounded by \(4(\lambda_{\text{max}} - \lambda_{\text{min}})\), and can be represented by 8 bits (to represent from −121 to +121). To prevent metric overflow or renormalization requirements, the modulo attribute of 2’s complement arithmetic is employed [Hek89]. This requires the path metric calculations to be done in 9-bit 2's complement arithmetic. Fig 4.5 show a block diagram of the general ACS unit. Fig 4.6 shows the simulated performance of the decoder with 6-bit quantized data and 9-bit 2’s complement arithmetic.

![Figure 4.5: Precision requirements of general ACS unit](image)

![Figure 4.6: Rate 6/8 code performance with 6-bit quantization and 9-bit path metrics](image)
4.4.3 Path Memory

The required path memory is determined by the length of the longest minimum distance events. With the \(1 - D\) channel and the zero disparity code of length 8, minimum distance events are less than 16 bits long as described in section 3.7.1. Fig 4.7 shows the simulated performance of the decoder with various path memory lengths above and below 16 bits with random source data. Below 16 bits, there is a distinct loss in the performance of the decoder due to non-merging paths especially at high SNR.

![Figure 4.7: Rate 6/8 code performance vs Path Memory](image)

4.4.4 Synchronization

One of the requirements of the time varying detector is the requirement of synchronization before the decoder begins working properly. The method of synchronization chosen is based on searching through a known preamble for a particular sequence. The decoder is fixed in state 0 which causes the decoder to act as a conventional PR4 decoder. A particular pattern was chosen to have a high probability of detection over the preamble sequence and shifted versions of itself. This type of scheme is suitable for disk drive applications where the presence of the sector preambles would be already known. Alternative schemes would be required in a tape drive. Fig 4.8 shows the performance of the system with the transfer of 1K Byte blocks of data and synchronization. It can be seen that the required coding gain of 3\(dB\) is achieved by the coded channel over the uncoded channel.
4.5 VLSI Prototype

A VLSI implementation of the rate 6/8 MSN code and detector described has been undertaken to determine the implementation performance and complexity. The 0.7um ES2 Eurochip standard cell process was targeted. The design consists of a MSN encoder, Viterbi detector and decoder. Additional test circuitry such as a pseudo random source bit generator, error rate counter and a $1 - D^2$ channel are also included. Fig 4.9 shows a block diagram of the prototype chip.

4.5.1 Transmit Section

Fig 4.10 shows a block diagram of the transmit section.

The transmit section is triggered by the external input go to send a block of 16896 user data bits. This allows 2KBytes per block and 64 Bytes for overhead. This data is preceded by a 64-bit preamble pattern, a 14-bit synchronization pattern and 32 further preamble bits. When not triggered a continuous preamble pattern is sent. The hand shaking of the input data is handled by a data request line. When the txdatareq line is asserted the external device must have the next data bit ready for the next clock rising edge. Fig 4.11 shows the block sequence and fig 4.12 shows the input data handshaking. The 6/8 encoder has been synthesized from table 4.1. The input bits are differentially precoded, encoded with the 6 to 8 encoder, interleaved for the PR4 channel and clocked out. The precoding enables the correct bits to be recovered irrespective of any 180 degree phase shifts in the channel. The
precoded data $b_k$ is determined from the user input data $a_k$ with the operation

$$b_k = a_k \oplus b_{k-2}$$

(4.2)

where $\oplus$ represents the exclusive-or operation.

4.5.2 Viterbi Detector and Receive Section

The receive section comprises of the Viterbi detector and decoding logic. The Viterbi detector operates on the 6-bit input samples to produce decisions on the data bits.
4.5.3 Viterbi Detector

In principle the Viterbi detector consists of branch metric calculations, ACS units and survivor path memories. The input signal has noiseless values of \{-2,0,+2\} and hence, three branch metrics should be calculated. The simplified branch metrics are calculated as

\[
\lambda_{-2} = -y_k \quad \lambda_0 = 1 \quad \lambda_{+2} = +y_k
\]  \hspace{1cm} (4.3)

Using these branch metrics, metric calculations are avoided and the received data samples are distributed to each ACS unit. Hence, when a branch metric of \(\lambda_{-2}\) is required to be added to a path metric, the current data sample is just subtracted. This avoids the requirement of multiple branch metrics with separate routing. In the design, the 6-bit 2’s complement number is scaled such that a sample value of 2.0 is represented as the decimal value of 24.

The time varying Viterbi detector consists of 6 ACS units whose operation depends on a 3-bit state word. When the search pin is asserted and released, the detector searches for the synchronization pattern in the input data. When this is found, the time varying trellis is synchronized to the data. The ACS units and path memory have registers that automatically operate on the appropriate interleave of the input samples to correctly decode the input samples. The ACS units are interconnected as shown in fig 4.13.
ACS Units

Each ACS unit has a certain operation based on the state of the detector. The different operations required by each ACS unit are selected by multiplexors and control logic. Fig 4.14 shows the configuration of the ACS unit for state **DK**. For this state, a multiplexor selects whether the path metric being considered is from **AI** or **DK**. A nand gate allows the metric update to be forced to a particular selection when this is required. Registers are also shown in the diagram. These implement the deinterleaving of the PR4 sequence. While the branch calculations are being performed on the even samples, the compare and select operations are being performed on the odd samples, and vise versa.

The 'Add' and 'Compare' operations of the ACS units are performed by 2’s complement adders and subtracters. The 9-bit path metrics have the 6-bit branch metrics added to them with carry select adders. The carry select adders operate as shown in fig 4.15. Each group of 3 bits is added to produce the sum if the carry in was a logic 0 and the sum if the carry in was a logic 1. A multiplexor then selects which one is output when the correct carry arrives. This form of adder was selected due to its good performance and ease of implementation. The 9-bit comparators were implemented as 9-bit subtracters in a similar manner.
**Speed Issues**

For high density magnetic recording, speed of operation is normally a very important consideration. In the design presented here, the only section of the logic that cannot be pipelined is the ACS units and hence, these are the ultimate speed bottle neck in the system. The interleaving of the PR4 channel samples requires that a complete Add Compare and Select (ACS) for a given sample is required every two samples. This allows the design of the ACS units as shown previously in fig 4.14.

The speed of the ACS unit is limited by the speed of a 9-bit adder and 2 multiplexors each clock cycle. Based on post route logic simulations and carry select adders, 75MHz operation should be typically achievable (50MHz worst case). If two independent paths for each interleave were implemented, higher speed would be achieved at the expense of an increase in area. The speed increase would be due to the elimination of the register between the add and compare section. This would allow some further optimization of the logic due to tighter coupling between them.

### 4.5.4 Path Memory

The path memory is implemented using register transfer method. All the registers in the path memory are clocked on the negative edge of the clock to reduce current spikes on the
positive edge of the clock. The path memory contains $6 \text{states} \times 16 \text{depth} \times 2 \text{interleaves} = 192$ register elements. Due to the time varying nature of the detector the path memory updates require a more complex scheme than in normal detectors. Table 4.5 lists the allowable updates for each state.

<table>
<thead>
<tr>
<th>State</th>
<th>Allowable updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG</td>
<td>BG AI</td>
</tr>
<tr>
<td>AI</td>
<td>BG AI CH</td>
</tr>
<tr>
<td>CH</td>
<td>DK CH FJ</td>
</tr>
<tr>
<td>DK</td>
<td>CH DK AI</td>
</tr>
<tr>
<td>FJ</td>
<td>FJ EL DK</td>
</tr>
<tr>
<td>EL</td>
<td>EL FJ</td>
</tr>
</tbody>
</table>

Table 4.5: Allowable path memory updates

The path memory is implemented as shown in fig 4.16. Each depth of the path memory has two cascaded flip flops to account for the PR4 deinterleaving.

**Synchronization**

Before codeword synchronization is achieved, the detector is operated in state 0. In this state, the detector operates as a conventional PR4 detector. A synchronization circuit detects the synchronization pattern and then switches over to the time varying trellis mode.
4.5.5 Receive Section

Fig 4.17 show a block diagram of the receive section.

![Receive Section Block Diagram](image)

Figure 4.17: Receive section block diagram

The receive section takes the data from the end of the Viterbi detector path memory, deinterleaves the odd and even bit streams, frames the data into 8-bit code words and decodes these into 6-bit words. This data is the precoded user data. The user data is
recovered by unprecoding and fed out serially in a similar manner to the input data. The user data \(a_k\) is determined from the precoded data \(b_k\) with the operation

\[
a_k = b_k \oplus b_{k-2}
\]

(4.4)

where \(\oplus\) represents the exclusive-or operation.

The user data appears on the data pin and the valid pin is asserted when the data is valid user data. The 8/6 decoder has been synthesized from table 4.1.

4.5.6 Support and Test Logic

Fig 4.18 shows the support and test features on the prototype chip. For ease of testing

![Diagram of support and test logic](image)

**Figure 4.18: Support and test logic**

of the device and error rate measurements, additional logic has been implemented on the prototype IC. Two Pseudo Random Bit Generators (PRBG) are implemented to allow random source data to be generated on chip. One is used as a source for the transmit section and the other is used is used to check for errors in the received data. Asserting the loadb pin causes both the contents of the transmit PRBG register to be loaded into the receive PRBG register, thus synchronizing them. Each PRBG has a 10-bit register and generates a sequence that repeats every \(2^{10} - 1\) bits[LM94]. The pin prngNOTtxdata selects whether the user data of the PRBG is used as the data source.

When the PRBG is used as the data source, the received data can be compared and bit errors identified. The error pin is asserted whenever such a bit error is detected. Each
error increments a 15-bit error counter. Asserting the **search** pin causes the error counter to be reset. The value of this counter can be read through an on-chip serial port. The **sclk** pin clocks the serial port. When the **sload** pin is asserted, the next **sclk** rising edge loads the serial port register with the error counter. The MSB of the register then appears on the pin **sdata**. Each subsequent rising edge of **sclk** with **sload** negated causes the serial port register data to be shifted out on the **sdata** pin.

In order to allow functional testing of the device, an internal $1 - D^2$ channel is available. A logic 0 on the **yinNOTchan** pin causes the transmit section data to be fed into the internal channel and the output of the channel to go to the Viterbi detector. In this way, the IC may be configured to generate random data, encode it and pass it through the internal channel and Viterbi detector. The received data is then decoded and compared to the original data. This allows the chip functionality to be quickly tested.

The received data samples are also fed to a loop calculation block that calculates a timing and gain error calculation from the data samples. These can be used for AGC and timing recovery. The gain error is calculated as

$$ g_{error} = \hat{y}_k (y_k - \hat{y}_k) $$

and the timing error is calculated as

$$ t_{error} = -y_k \hat{y}_{k-1} + y_{k-1} \hat{y}_k $$

where $y_k$ is the data sample value and $\hat{y}_k$ is a threshold detected value

$$ \hat{y}_k = \begin{cases} 
2 & y_k \geq 1.0 \\
0 & -1.0 < y_k < 1.0 \\
-2 & y_k \leq -1.0 
\end{cases} $$

Other test functions are provided with a number of test pins.

**pr4NOTmsn**: A logic 1 on this pin forces the detector to remain in the PR4 mode to allow PR4 error rates to be measured.

**testerrcnt**: This pin enables a test mode on the 15-bit error counter by breaking it up into 5-bit counters.

**testup,testlo**: These pins allow the transmit and receive counters to be tested. Asserting each one reduces the number of bytes transmitted for test purposes.

**reset**: This is a global reset signal to ensure that the IC may be reset into a known state for testing purposes.
4.5.7 Design Methodology

The functions and specifications for the chip were developed and verified through C language programs. These programs provided performance data and stimulus data for logic simulations of the hardware.

The 6/8 encoder and 8/6 decoder were synthesized from Verilog hardware descriptions with the Cadence HDL synthesizer using a 2.4um Mitec Library. The resulting schematics were then transferred to the ES2 0.7um process. All the remaining circuitry was entered by schematics based on the ES2 0.7um standard cell library[199]. Synthesizer libraries were not available for the ES2 process at that time.

Logic simulations were carried out using the Verilog XL simulator. Hierarchical place and route was not available so the top level design was flattened and placed and routed automatically\footnote{Some manual intervention was required to correct some placement problems} by the Cadence place and route software. This achieved routing of all nets in the design.

4.5.8 Design for Test

The prototype system has an on chip Pseudo Random Bit Generator (PRBG) and this is used to send a test data sequence through the encoder, Viterbi detector and decoder logic. The decoder sequence is then compared to the generated sequence to verify correct operation of the complete system. A small amount of additional test circuitry is also implemented to break long counter chains. The selected test vectors toggled 93% of all the circuit nodes. The use of this built in test logic provides a good functional test without external test equipment. The built in bit sources and error counter allows measurements of bit error rates to be readily achieved with an external channel and noise source.

4.5.9 Prototype IC Specifications

The prototype chip has 47 used pins and is packaged in a 48pin DIL package for convenience of testing. The prototype chip has about 11000 nand gate equivalents based on the ES2 0.7um standard cell library and an area of 3.63mm by 3.27mm including IO pads. Fig 4.19 shows the prototype IC layout with the areas devoted to the major sections outlined. Appendix E shows some representative schematics from the design.

4.6 Prototype Measurements

Prototype ICs were fabricated through the EUROCHIP services and 10 packaged devices were received. The functional tests and performance measurements done are now
4.6.1 Functional Test

The functional testing of the devices has been simplified by the built-in test features of the device. The IC is configured as in figure 4.20. In the test mode the transmit section signal is fed into the receive section. On the receipt of a negative transition on the go pin a synchronization pattern and a frame of data is sent appearing on the Iwrite pin. This data is also fed into the internal $1 - D^2$ channel and into the Viterbi detector. The transmit section busy line is asserted in response to the start of the data transmission and is negated when the transmitter is completed. The txdatareq pin can also be observed to pulse for each 12 bits of input data requested. During data transmission, the Iwrite pin can be observed to toggling with the write data while at all other times a $\{1,1,0,0,1,1,0,0 \ldots\}$ pattern should be present.
The receive section should activate the **valid** pin for each 12-bit data word received. This can be observed to indicate that the receive section has synchronized and is receiving data. The **error** pin can be observed to indicate whether errors are occurring in the received data. With the **yinNOTchan** pin held high, there should be no errors as the correct data samples are fed through the internal channel. With the **yinNOTchan** pin held low, the samples are received from the pins **y[5:0]** and numerous errors can be observed on the **error** pin. The serial port may be read using the pins **sclk**, **sload**, **sdata** to check that the error counter is working.

These simple tests indicate that all the main data paths of the IC are operational and functioning correctly.

![Diagram](image)

**Figure 4.20: Functional test circuit**

The 10 devices were checked for functionality using these tests. Of these, 9 were found to be fully functional. One device had a fault in the **error** counter section. Visual inspection of this device revealed a defect on the die.

### 4.6.2 Speed of Operation Measurements

The maximum speed of operation can be determined with a similar setup to the functional test circuit of figure 4.20. In this case random data is generated internally, encoded
and sent through an internal $1 - D^2$ channel. The channel output is sent through the Viterbi detector and decoded. The decoded bits are then compared with the initial random data and any errors indicated on the error pin.

The clocking rate of the device can be increased until the error line begins to go high. This indicates that the internal loop back is not functioning correctly. The usable operation of the chip would be less than this maximum test frequency.

The maximum test frequency of the device was measured at a number of supply voltages for one device at room temperature. The measurement results are shown in graph of fig 4.21.

![Graph](image)

**Figure 4.21: Maximum Test Freq vs Supply Voltage**

4.6.3 Supply Current

The supply current as a function of frequency was measured at a supply voltage of 5V and the measurement results are shown in fig 4.22. This is the supply current taken by the device alone in the test mode with both the transmit and receive sections operating together.

4.6.4 Design Performance and Verification Test

The performance of the IC as a transmit encoder and Viterbi receiver is tested with the circuit of fig 4.23.

The output write level signal is formed into a PR4 sequence by subtracting a delayed version of itself. A noise source adds Gaussian noise to the PR4 signal and this is digitized with a 6-bit flash ADC. These 6-bit samples are then used by the Viterbi decoder in the MSN chip. The reference voltage of the ADC is adjusted to properly scale the samples. This circuit has been constructed and the control lines and serial port are controlled by a
PC through the parallel port. This allows the PC to strobe the device causing a block of random data to be transmitted and received. The serial port on the chip is then used to read the error counter register value back into the PC. Error rates can then be measured by adjusting the noise source and using the error count from the serial port. Using this method the resulting measurements indicated in fig 4.24 were taken. These BER measurements show that the MSN code provides the expected 3dB decrease in bit error rate.

The effect of ADC resolution was confirmed by measuring the error rate with a fixed amount of noise and varying the number of ADC bits fed to the MSN chip. The results of this measurement is shown in fig 4.25. This shows that 4 bits of resolution provides virtually all the available gain as predicted by simulations.

4.7 Conclusions

A VLSI implementation of a rate 6/8 MSN encoder, decoder and Viterbi detector has been designed based on the zero disparity codes presented in chapter 3. High level simulations were used to verify the performance of the code and detector. They were also used to determine the resolution of the calculations required for the implemented detector.

A prototype IC was designed using the ES2 0.7um standard cell process. The prototype chip implemented the encoder, decoder and Viterbi detector. Additional test and measurement logic was also included for test purposes. These IC’s were fabricated through the Eurochip services.

Based on the received prototype IC’s, it has been determined that the devices are functional. Basic functional tests show that the chips perform as in logic simulations. Of the 10 chips received, 9 were functional and gave the required MSN gain vs PR4 while 1
device failed to function due to a defect on the die.

Speed tests on the devices show that they can readily be clocked in excess of 100MHz in the test mode. This indicates that they are consistent with the expected 75MHz typical operation and 50 MHz worst case operation predicted by simulations.

The BER performance of the device has been measured and shows that the device is working as designed. The 3dB gain due to the use of the MSN code has been measured in the presence of Gaussian noise. The effect of ADC resolution has also been measured and verified to be consistent with simulations.

This implementation verifies the feasibility of implementing matched spectral null codes using time varying trellises.
Figure 4.24: BER Measurements

Figure 4.25: Performance of MSN with various ADC resolutions
Chapter 5

Distance Lengthening Codes

5.1 Introduction

In this chapter, codes that eliminate certain minimum distance events are proposed. It is shown that such codes can provide an improvement in performance on a PR4 channel due to the effects of noise correlation. First, parity check codes are shown to achieve the desired gain but suffer from poor run length constraints. Modulo RDS codes are then developed which achieve the same gains but with a more efficient detector and better run length properties. An example rate 16/18 code is developed in detail with consideration given to implementation complexity, run length constraints, path memory requirements and the effect of error events. These codes are based on eliminating short minimum distance error events and shall be called Distance Lengthening (DL) codes.

5.2 Losses Due to Noise Correlation

While the PR4 partial response channel is a good target for the magnetic recording channel [TP87], considerable equalization is required especially at higher recording densities. This equalization causes correlation of noise appearing at the Viterbi detector and this can reduce the performance of the detector. For the PR4 Viterbi detector, the probability of error was calculated in chapter 2, (eqn 2.83) as

\[ P_e(PR4) = \sum_{i=1}^{\infty} \frac{2i}{2^i} Q\left( \frac{\sqrt{2}}{\sqrt{R_n(0) - R_n(2i)}} \right) \quad (5.1) \]

It can be seen from this expression that the error performance of the detector can be degraded by the effect of noise correlation. Fig 5.1 shows the normalized autocorrelation of the noise on the PR4 detector input at recording densities of \( \rho = 1.5 \) and \( \rho = 2.5 \) bits per \( PW_{50} \) based on the model presented in chapter 2.

It can be seen that there is a significant amount of correlation between closely spaced data samples. Due to the interleaved nature of the PR4 channel, correlation between
adjacent samples does not effect the error rate as described in eqn 5.1. Only correlation between even spaced samples degrades the error rate. In the case of a recording density of \( \rho = 1.5 \) as in fig 5.1(a) the correlation between samples a distance 2 apart is \(-0.33\) and between samples a distance 4 apart is \(-0.078\). The other even distances are not so significant.

The loss in error rate due to correlation of samples a distance 2 apart is determined as

\[
Q\left(\frac{\sqrt{2}}{\sqrt{R_n(0)/1.0 - (-0.33)}}\right) \tag{5.2}
\]

Here, the correlation causes a loss of \(1.23\) dB in effective signal to noise while the loss due to correlation between samples a distance 4 apart is \(0.3\) dB. At a recording density of \(\rho = 2.5\), the losses are \(0.9\) dB and \(0.7\) dB.

Hence, if it were possible to eliminate the error events that are subject to the effects of noise correlation, it would be possible to gain an increase in effective SNR for a given channel. It is proposed here to achieve this through employing a source coder that applies some structure to the recorded signal. This structure can then be taken into account in the detection process. The performance of such codes on the PR4 channel can be calculated based on the the probabilities of minimum distance error events. The probability of error for the PR4 channel was calculated as in eqn 5.1.

If minimum distance events of length 2 and 3 are eliminated, then the summation can be taken from events of length\(^1\) 4 with the resulting probability of error being

\[
P_{\text{error}} = \sum_{i=3}^{\infty} \frac{2i}{2^i} Q\left(\frac{\sqrt{2}}{\sqrt{R_n(0) - R_n(2i)}}\right) \tag{5.3}
\]

\(^1\)Events of length \(l\) correspond to \(i + 1\) in the summation

111
Using this expression in the model developed in chapter 2 allows the performance of such codes to be evaluated compared to the uncoded PR4 and EPR4 channels. Fig 5.2 shows the compared performance under white noise conditions. The required SNR for a bit error rate of $1 \times 10^{-6}$ relative to the PR4 channel is plotted. This shows that with the minimum distance events of length 2 and 3 eliminated, 1.25dB of improved noise performance can be obtain at a recording density of 1.5 and over 1dB improvement up to a recording density of 2.4.

![Figure 5.2: Comparison of performance](image)

### 5.3 Minimum Distance Errors on the Dicode Channel

Fig 5.3 shows the trellis diagram for a non-precoded channel. All minimum distance error events have a minimum distance of $2\sqrt{2}$, and as shown in section 2.7.5, have the form

$$e_k = \pm \{2, \ldots, -2\}$$  \hspace{1cm} (5.4)

An error event of length $l$ will have $l - 2$ zeros between the +2 and -2. By comparing with the trellis diagram, it can be seen that each error event of length $l$ will have $l - 1$ consecutive bit errors and each bit error will be the same i.e all $1 \rightarrow 0$ or all $0 \rightarrow 1$. It can also be seen that a minimum distance error event of length $l$ requires $l - 2$ consecutive 0 channel samples for the event to be supported.

In order to design a code to eliminate minimum distance error events of certain lengths, a source coding scheme is required that can detect the errors that would result for those
error events. Hence to avoid error events of length 2, a source code is required that can
detect a single bit error. To detect minimum distance error events of length 2 and 3, a
source code is required that can detect a single bit error and 2 consecutive identical bit
errors. This could be done through the use of block codes or convolutional type codes. The
code must also provide run length constraints to avoid long runs of consecutive symbols
which do not provide sufficient information for timing and gain recovery.

5.4 Parity Codes to Reduce Noise Correlation

Parity codes provide the most straight forward method of detecting single bit errors and
could be used to avoid error events of length 2. In order to eliminate minimum distance
errors of length 2 and 3 in the non-precoded Dicode channel, a code is required that can
detect single bit errors and errors in two consecutive bits. The most straight forward
method of doing this is to use an interleaved single parity check code. An odd parity check
code can be described by a trellis as shown in fig 5.4 for a rate 8/9 code.

![Figure 5.4: Rate 8/9 odd parity trellis](image)

Two of these codes can be interleaved to form the trellis shown in fig 5.5 which has 4
states and a rate of 16/18.

When the Dicode channel is combined with the trellis of fig 5.5, an 8-state trellis results.
This is a trellis that varies every second sample. Fig 5.6 shows a single section of the trellis.
After 18 samples, the trellis should converge to one of two possible states EE, 0 or EE, 1.

Decoding of the received data can be accomplished using a time varying Viterbi detector
on the trellis in fig 5.6. At the end of 18 data samples, all the states except EE, 0 and EE, 1
are discarded and the path metrics and survivor paths for these states are transferred to
states $OO,0$ and $OO,1$ respectively. The rate of a single error detection parity check code is $\frac{n-1}{n}$ for a block length of $n$.

5.4.1 Performance of Parity Codes

In the case of a single bit detection parity check code, any odd number of errors can be detected. The interleaved parity check code can also detect certain error patterns in addition to the single and double bit error and hence the performance should be better than that predicted in fig 5.2.

The performance of each of these codes was simulated using the computer model developed in chapter 2. Error rates were simulated for a range of recording densities and the results are plotted in fig 5.7. This plot shows the effective SNR gained through the use of the parity codes with a PR4 channel being represented as a reference at $0dB$. The
single bit detection code was rate 8/9 odd parity and the interleaved code was rate 14/16 interleaved odd parity².

![Graph showing the performance of parity codes with the PR4 channel as the 0dB reference](image)

Figure 5.7: Simulated performance of parity codes with the PR4 channel as the 0dB reference

The single bit error detection parity code achieves a gain of almost 0.75dB over the base PR4 channel at recording densities of 2.0 bits per PW50 and almost 1dB at lower densities. The interleaved parity check code achieves over 1.25dB gain at recording densities below 2.25 bits per PW50. The EPR4 channel exceeds the performance of the interleaved parity code only at densities above 2.5 bits per PW50. This confirms the potential gain from such codes and shows they can extend the performance of PR4 based channels without going to an EPR4 channel.

5.4.2 Run Length Properties of Parity Codes

While parity codes can achieve a high rate, their use has the disadvantage of long run lengths of identical symbols. This causes problems for timing recovery and AGC loops. Consider a parity code with n bits. For even parity the bit sequence \{0,0,\ldots,0\} is a valid code word and cannot be eliminated because all codewords are required for a rate \(\frac{2n+1}{n}\) code. For odd parity the \{1,1,\ldots,1\} is a valid code work if \(n\) is odd. If \(n\) is even, then odd parity requires at least one different bit in each codeword. The code would then allow a sequence with \(2n-2\) consecutive bits of identical bits. An interleaved version could allow \(2(2n-2)\)

²Rate 14/16 was used to allow a path memory of two codewords fit in a single 32 bit integer for programming purposes
consecutive sequences of identical bits. For a rate 14/16 interleaved code \( n = 8 \) and the maximum possible run length of 0’s would be 28. However, this is for the Dicode channel and operation on the PR4 channel requires two interleaved Dicode channels. Hence, the maximum possible run length would again increase. This could double the possible run length of identical to 56 identical bits. These excessively long runs of identical bits result in channel samples of 0 at the PR4 channel output and hence can cause problems for timing recovery and AGC control.

5.4.3 Complexity of Implementation

The source encoder and interleaver for such codes are very simply implemented. The user data can be recovered from the received codewords by deinterleaving and stripping off the parity bits.

The Viterbi detector requires 4 states for the single parity code and 8 states for the interleaved parity code. These have to operate at half the channel rate which reduces the complexity compared to an EPR4 detector which must operate at the full channel rate. The detectors need to be synchronized to the incoming data similar to the MSN chip described in chapter 4. The path memory would need to be of the order of 2 codewords to prevent catastrophic sequences as described in section 3.7.

5.5 Modulo RDS Codes

Although parity codes as previously described provide the required coding gain, it was shown that they suffer from long run lengths of identical bits. An alternative coding scheme based on modulo RDS codes is proposed which achieves better run length constraints.

As described in chapter 3, a zero disparity code used on a Dicode channel results in a code with increased minimum distance. This is the case because any minimum distance event causes a group of consecutive identical bit errors. When such an error event occurs, the zero disparity of the code is violated and cannot be restored unless a minimum distance error event of opposite polarity occurs or a non minimum distance error event occur.

This property can also be used to eliminate error events of certain lengths. This is achieved here by using codewords that have a fixed RDS modulo \( M \), with \( M \) being suitably chosen. In the present context minimum distance error events of length 2 and 3 can be detected and discarded by considering codewords with a \( |RDS|_3 \) of 0. Let the source codeword be \( C = \{c_0, \ldots, c_{N-1}\} \) with \( c_k \in \{1, -1\} \). Since it is a codeword

\[
|RDS(C)|_3 = \sum_{k=0}^{N-1} c_k = 0
\]

(5.5)
A minimum distance of length 2 causes one bit error of $1 \to -1$ or $-1 \to 1$. In the case of a $1 \to -1$ the received codeword $\hat{C}$ will have a $|RDS|_3$ of

$$|RDS(\hat{C})|_3 = -2 + \sum_{k=0}^{N-1} c_k \bigg|_3 = 1$$

(5.6)

and hence is not a codeword. The error $-1 \to 1$ causes the received codeword $\hat{C}$ will have a $|RDS|_3$ of

$$|RDS(\hat{C})|_3 = 2 + \sum_{k=0}^{N-1} c_k \bigg|_3 = 2$$

(5.7)

and hence is not a codeword either.

In the case of a minimum distance error of length 3, there are two consecutive bit errors, both $1 \to -1$ or both $-1 \to 1$. In the case of both errors being $1 \to -1$ the received codeword $\hat{C}$ will have a $|RDS|_3$ of

$$|RDS(\hat{C})|_3 = -2 - 2 + \sum_{k=0}^{N-1} c_k \bigg|_3 = 2$$

(5.8)

and hence is not a codeword. Similarly, in the case of both errors being $-1 \to 1$, the received codeword $\hat{C}$ will have a $|RDS|_3$ of

$$|RDS(\hat{C})|_3 = 2 + 2 + \sum_{k=0}^{N-1} c_k \bigg|_3 = 1$$

(5.9)

and hence is not a codeword either.

Consider minimum distance events of length $l$ on the Dicode channel. Such error events cause $l-1$ consecutive identical errors. Each error causes the $|RDS|_3$ to be increased by 2 or decreased by 2. $l-1$ consecutive identical errors causes an increase or decrease by $2(l-1)$. To cause a codeword $C$ with $|RDS|_3 = 0$ to be changed into another valid codeword $\hat{C}$ requires

$$|RDS(\hat{C})|_3 = |RDS(C) \pm 2(l-1)|_3 = |\pm 2(l-1)|_3 = 0$$

(5.10)

as $|RDS(C)|_3 = 0$. Hence $2(l-1) = 3k$ for $k = \{1, 2, \ldots\}$, and the possible minimum distance error events are of length $\{4, 7, 10, \ldots\}$. The bit error rate probability can then calculated as

$$P_e(|RDS|_3) = \sum_{i=3,6,...}^{\infty} \frac{2i}{2^i} Q \left( \sqrt{\frac{2}{R_{n_t}(0) - R_{n_t}(2iT)}} \right)$$

(5.11)

Fig 5.8 shows a trellis describing a code with a $|RDS|_3 = 0$. The code must be a block code starting with a RDS value of 0 and terminating with an identical RDS value of 0.

When combined with the trellis of a Dicode channel, the 6-state trellis of fig 5.9 results. Each state is labeled with $|RDS|_3$, last input. This trellis should terminate in the states 0,0 or 0,1 at the end of each code word.
5.5.1 Rates of Modulo RDS Codes

Consider a codeword of length \( n \). Let there be \( m \) symbols of +1 and hence \( n - m \) symbols of −1. The total RDS of the codeword is \( m - (n - m) = 2m - n \). Given the RDS modulo \( M \) is zero as required then the number of code words can be calculated. Given \( m \) 1’s, the number of ways of placing \( m \) 1’s in \( n \) slots is

\[
\frac{n(n-1)\ldots(n-m+1)}{m!} = \frac{n!}{m!(n-m)!}
\]

(5.12)

Hence, the total number of codewords can be calculated as

\[
N_{\text{codewords}} = \sum_{m=0}^{n} \frac{n!}{m!(n-m)!}
\]

(5.13)

Table 5.1 tabulates the rates for various values of \( n \) which may be also an odd number which is not the case with zero disparity codes (section 3.3.3). The table shows the rate of the code that could be theoretically achieved and the block rate that can be achieved by encoding a fixed number of bits per block.

5.5.2 Encoding and Run Length Properties of Modulo RDS Codes

From table 5.1, it can be seen that a rate 16/18 code can be implemented. The simplest method of encoding the input 16 bits would be to set the first 16 bits of the codewords to
the 16 bits to be coded. The $|RDS|_3$ of these 16 bits is then calculated which will result in a value of 0, 1 or 2. The last 2 bits can then be filled in to bring the $|RDS|_3$ value of the codeword to 0. Table 5.2 shows the required bits to be appended.

\[
\begin{array}{c|c|c|c}
16 \text{ bits} & |RDS|_3 & \text{bit 17} & \text{bit 18} \\
\hline
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
2 & 0 & 0 \\
\end{array}
\]

Table 5.2: Appending bits to realize codewords with $|RDS|_3 = 0$

This encoding scheme is similar to parity coding and has similar characteristic of possible long run lengths of identical symbols. Consider the all 0 input data word of length 16 bits. The $|RDS|_3$ of this is 2 which requires the bits 00 to be appended. This allows the possibility of an infinite run of 0’s which is unacceptable for timing and AGC recovery.

However, looking back to table 5.1, it can be seen that there are 87382 code words meeting the constraint $|RDS|_3 = 0$ while only 65536 are required to code 16 bits. This allows the possibility of eliminating 21846 of the worst case run length codewords. While this is possible in principle, an encoding table of $2^{16}$ words of length 18 and a decoding table of $2^{18}$ words of length 16 would be required. This would consume an excessive amount of area in any VLSI design. A compromise between the simple encoding/decoding with long run lengths and large tables can be arrived at. An example is given in the next section.
5.5.3 Example Rate 16/18 $|RDS|_3 = 0$ Code

Consider partitioning the 16 input data bits into 10 bits and 6 bits. The $|RDS|_3$ of 10 bits is calculated resulting in 0, 1 or 2 and will contribute to 10 bits of the codeword. This leaves 6 input bits and 8 codeword bits. By enumerating sequences of length 8, it can be shown that 86 have a $|RDS|_3$ of 0, 85 have a $|RDS|_3$ of 1 and 85 have a $|RDS|_3$ of 2. By searching through these 8 bit sequences it is possible to find 64 from each group that have transitions in both the first four bits and the second four bits. Hence, codewords can be constructed as follows with the input data being $d_0, \ldots, d_{15}$ and the codeword being $c_0, \ldots, c_{17}$. The tables of 8 bits sequences are denoted $tab_x$ with $x$ being the $|RDS|_3$ of each sequence of the table. $tab_x[k]$ is the $k$ sequence of the table.

- Calculate $|RDS|_3$ of $d_0, \ldots, d_9$ as $RDS_{10}$
  - if $RDS_{10} = 0$ let $s_0, \ldots, s_7 = tab_0[d_{10}, \ldots, d_{15}]$
  - if $RDS_{10} = 1$ let $s_0, \ldots, s_7 = tab_2[d_{10}, \ldots, d_{15}]$
  - if $RDS_{10} = 2$ let $s_0, \ldots, s_7 = tab_1[d_{10}, \ldots, d_{15}]$
- form the final codeword as

$$c_0, \ldots, c_{17} = d_0, \ldots, d_4, s_0, \ldots, s_3, d_5, \ldots, d_9, s_4, \ldots, s_7 \quad (5.14)$$

As each of the sequences $s_0, \ldots, s_3$ and $s_4, \ldots, s_7$ have transitions, the worst case run length of identical symbols is 11 symbols. If these are interleaved with one interleave shifted by 4 symbols, then the worst case run length of identical bits is 14. When passed through a PR4 channel which has a response of $1 - D^2$, the worst case run length of zero samples will be 12 zeros between a $+2$ and $-2$ sample. There will also be 2 or more non-zero samples after the worst case run length. This is considerable better than that achievable with parity type codes. Full details of a possible implementation of such a code is presented in section 5.8.

5.5.4 $RDS$ Modulo 3 Codes Path Memory Requirements

The depth of path memory required is an important parameter in the design of a Viterbi detector for VLSI implementation. In this section, a bound on the path memory required for a $RDS$ modulo 3 code is calculated. The Viterbi detector for the code consists of 6 states labeled with the $|RDS|_3$ value from the set $\{0, 1, 2\}$ and the last channel input bit from the set $\{0, 1\}$. The trellis was shown in fig 5.9 of section 5.5. Each branch was labeled with the current input bit from the set $\{0, 1\}$ and the channel output symbol from the set $\{-2, 0, 2\}$. The following properties of the trellis and code can be readily observed:
1: The state \((R, 0)\) has two outputs corresponding to input bits of 1 and 0. With an input bit of 1 the next state is \((|R + 1|_3, 1)\) with an output symbol of +2. With an input bit of 0 the next state is \((|R - 1|_3, 0)\) with an output symbol of 0. Similarly, the state \((R, 1)\) has two outputs. With an input bit of 1 the next state is \((|R + 1|_3, 1)\) with an output symbol of 0. With an input bit of 0 the next state is \((|R - 1|_3, 0)\) with an output symbol of -2.

2: At the end of each code word the encoder is known to be in states \((0, 0)\) or \((0, 1)\).

3: Whenever a transition in the input data occurs, there is an output symbol of +2 or -2.

Consider an error event starting at time index \(k\) and at the state \((R, 0)\) (the case of \((R, 1)\) will be symmetrical) \(R \in \{0, 1, 2\}\). The path with a 1 input will lead to \((|R + 1|_3, 1)\) with an output symbol of +2. The path with a 0 input will lead to \((|R - 1|_3, 0)\) with an output symbol of 0. The squared distance between these two paths is 4 and the time index is \(k + 1\).

The state \((|R + 1|_3, 1)\) can only have outputs 0 or -2 and the state \((|R - 1|_3, 0)\) can only have outputs 0 or +2. Thus, the only way the distance of 4 can remain is if both outputs are 0. This requires that the new states are \((|R + 2|_3, 1)\) and \((|R - 2|_3, 0)\) respectively at time index \(k + 2\). The same is true at time index \(k + 3\) when the paths end in states \((|R + 3|_3, 1)\) \((|R|_3, 1)\) and \((|R - 3|_3, 0)\) \((|R|_3, 0)\). At this point the two paths can converge if \((|R|_3, 1)\) receives an input bit of 0 causing an output of -2 and \((|R|_3, 0)\) receives and input bit of 0 causing an output of 0. The two paths then converge to the state \((|R - 1|_3, 0)\) and the squared distance between the two paths is 8. This occurs at the time index \(k + 4\) and represents a minimum distance event of length 4. They can also both converge to the state \((|R + 1|_3, 1)\). The only other possibility is channel symbols of -2 and +2 requiring a squared distance of \(4 + 16 = 20\).

If the paths do not converge at the time index \(k + 4\), then a squared distance of 4 remains. The cycle can continue in this manner if contiguous streams of 0's occur at the channel output. Convergence with the squared distance of 8 can occur at time index \(k + 1 + 3n\), \(n = 1, 2, \ldots\) when the states have the same \(|RDS|_3\) value and different previous bit values.

If the encoder is designed to constrain the run length of 0's at the channel output, which is required for AGC and timing recovery purposes, then this places an upper limit on the lengths of minimum distance events. If the maximum run length of channel 0's is \(L\), then the minimum distance error events must have a length \(3n + 1 \leq L + 2\).

When the time index is not at \(k + 1 + 3n\), \(n = 1, 2, \ldots\) and the squared distance is 4 the states can be \((|R + 1|_3, 1)\) and \((|R - 1|_3, 0)\) or \((|R + 2|_3, 1)\) and \((|R - 2|_3, 0)\) as shown.
above. Take the case of \((|R + 1|_3, 1)\) and \((|R - 1|_3, 0)\) and let the output from \((|R + 1|_3, 1)\) be \(-2\) with a channel input of 0. The output from \((|R - 1|_3, 0)\) is 0 with a channel input of 0. This increases the squared distance to 8. The new states are \((|R|_3, 0)\) and \((|R - 2|_3, 0)\). These states have identical previous bit values and different \(|RDS|_3\) values. The same can be shown for the case of channel outputs of 0 and 2 and for the case of \((|R + 2|_3, 1)\) and \((|R - 2|_3, 0)\).

With the same previous bit values but different \(|RDS|_3\) values, the paths can have identical sequences of any symbol values and these gain no distance between them. These are catastrophic sequences. However, at the end of each code word, the trellis is known to be in states \((0,0)\) or \((0,1)\) and hence, one of the paths can be eliminated at this time. Otherwise an additional sequence difference must occur and this will increase the squared distance of the paths to 12 or more.

Some example error events are shown in fig 5.10, 5.11 and fig 5.12. In fig 5.10 a minimum distance error event of length 4 is shown. Fig 5.11 shows an error event starting with \(\pm\{2,0,-2,0,\ldots\}\) that is eliminated at the code word boundary due to the trellis converging. Fig 5.12 shows an event starting in one codeword and crossing the codeword boundary with an accumulated distance of 4. The event can then extend in the next code word. The error event can converge with a distance of 8 as a minimum distance event. Otherwise, it gains a distance of 8 and extends as a catastrophic sequence as shown. In this case it would be eliminated at the next codeword boundary. It can be noted that this boundary crossing error event can start, at worst case, at the last transition in the previous codeword. This fact is used in the design of a rate 16/18 code described in section 5.8.

![Diagram](image_url)

**Figure 5.10:** Example minimum distance error event \(\pm\{2,0,0,-2\}\)

The path memory requirements of a code can be calculated based on the location of transitions and run length constraints. This will be done later for a specific code implementation.
Figure 5.11: Example error event $\pm\{2,0,-2,\ldots\}$ being eliminated at codeword boundary

Figure 5.12: Example error event crossing codeword boundary

5.5.5 Performance of Modulo RDS Codes

The plot of fig 5.13 shows the simulated performance of a rate 16/18 $|RDS|_3 = 0$ code and theoretical performance predicted by eqn 5.11 based on minimum distance error events. The 0dB line represents the performance of the uncoded PR4 channel. This simulation verifies that the modulo RDS code performs as expected and can provide 1.25 dB of gain up to a recording density of 2.0.

5.5.6 Complexity of Implementation

The complexity of implementation of a modulo RDS code as described above is quite reasonable when compared to that of a full 8 state EPR4 detector. The encoder and decoder could require significant silicon areas if lookup tables are used. Logic implementations may reduce the required area especially as high speed operation is not required. Section 5.8 describes a detailed encoder and decoder that may be readily implemented with moderate logic complexity. The $|RDS|_3 = 0$ code has 6 states and is only required to run at half the channel rate. This is considerably favorable compared to an 8 state EPR4 detector running at the full channel rate in terms of silicon area and power dissipation. The path memory
Figure 5.13: Performance of $|RDS|_3$ code relative to the uncoded PR4 channel

required for convergence of all minimum distance events is of the order of two codewords. The actual length of path memory required will depend on details of the code. As the trellis converges to two possible states at the end of each 18-symbol codeword, only two paths are required for the last 18 bits. Because the trellis converges to two possible states at the end of each 18-symbol codeword, the Viterbi detector is required to be synchronized to the input symbol stream. This requires some additional logic in the detector.

5.6 Error Events

In the case of the PR4 channel each minimum distance error event of length $l$ causes $l - 1$ adjacent bit errors. With precoding this changes to 2 error bits per minimum distance error event with $l - 2$ correct bits between the errors. These effects were taken into account in previous expressions for bit error probabilities through a term multiplying the $Q()$ function, as in eqn 5.1.

However, in a practical PR4 channel with 8/9 (0,4/4) coding, each minimum distance error can be of length 4 in a given interleave and with precoding this can cause 2 bit errors a distance 4 bits apart. When deinterleaved, these errors can be 8 bits apart and hence may exist in two different 9 symbol codewords. As there is not a simple mapping between each 9-bit code word and the corresponding 8-bit byte, a single error may cause from 0 to 8 of the bits in the user data byte to be in error. Hence, a single error event can cause errors in two adjacent bytes.

For this reason higher level error correcting codes are often of the symbol correcting
type such as Reed Solomon (RS) codes which correct errors in terms of m-bit symbols as opposed to individual bit errors[LC83]. In the case of the PR4 channel with 8/9 (0,4/4) coding described above, a reasonable error correcting scheme would be an interleaved RS scheme with 8-bit symbols. If each RS code could correct t symbol errors, then t minimum distance error events could be corrected as each minimum distance error event could at most cause errors in one byte in each interleave. An RS code with 8-bit symbols has a codeword with $\leq 2^8 - 1$ symbols. As typical sector sizes on rigid hard disk would typically be a multiple of 512 bytes, more than two interleaved codes would be required. In this case, the error performance of the system is dependent on the probability of error events occurring as opposed to the probability of bit errors.

5.6.1 Error Events with Modulo 3 RDS Codes

In section 5.5, minimum distance error events for the RDS modulo 3 codes were shown to be of length $l = \{4, 7, \ldots \}$. Hence, each minimum distance error event gives rise to more bit errors than the PR4 minimum distance error events. This results in a higher probability of bit errors.

![Figure 5.14: Performance of RDS modulo 3 code in terms of error events](image)

The probability of error events has been simulated on the Lorentz channel model. Fig 5.14 shows the resulting plot with the SNR based on error events rather than bit errors. The PR4 channel is represented as the 0dB reference. In this case, the RDS modulo 3 code performs better than on a bit error comparison. The gain in SNR terms is about
1.6dB over the range of interest. With a suitable higher level symbol correcting code, the probability of error events is more important than the probability of bit errors if multiple bit errors result from each error event.

5.7 Comparison with MLSD detector

Fig 5.15 shows the simulated performance of the uncoded PR4 channel, the RDS modulo 3 code and the MLSD bound from chapter 2. The plot shows the required channel SNR for a bit error rate of $10^{-6}$ with equal media and electronics noise. It can be seen that the RDS modulo 3 performs almost as good as MLSD up to a recording density of 2 bits per $PW_{50}$. This almost optimum performance is achieved by coding and hence, is not a fair comparison as the MLSD performance is for uncoded data. However, as any practical channel requires a line code for timing, AGC and equalizer adaption purposes, some rate loss will always be required.

Thus, the RDS modulo 3 code can achieve a performance close to optimum detection at moderate recording densities in a practical system.

![Graph showing the comparison of RDS modulo 3 code and MLSD](image)

**Figure 5.15: Comparison of RDS modulo 3 code and MLSD**

5.8 Detailed Rate 16/18 Encoder/Decoder

In this section, details of an encoder and decoder for a rate 16/18 RDS modulo 3 code is presented. The code design is developed to meet the following requirements.

**RDS** The RDS modulo 3 of the code should be zero for all codewords to achieve the benefits in error rate performance described previously.

126
Run Length Constraints The code should prevent excessively long strings of 0’s on the output of the PR4 channel. This is to prevent problems with timing recovery and AGC operations.

Channel Inversion If the channel polarity is inverted then the decoded user data bits are inverted. This requires that if a data word \( C \) is mapped to the coded sequence \( S \) then the inverted data word \( \overline{C} \) is mapped to the coded sequence \( \overline{S} \).

In addition, the code is designed to limit the propagation effects of minimum distance error events. When a minimum distance error event occurs, some of the channel bits at the output of the Viterbi detector will be in error. Depending on the number of errors and the position of the errors, it would be possible that two or more codewords would be incorrectly decoded which corresponds to 4 bytes of user data. The code presented here prevents minimum distance error events from causing errors across boundaries and limits the errors caused by a single minimum distance event to two adjacent bytes of user data.

5.8.1 Encoder

The encoder takes 16 bits of input data \( \{b_0, \ldots, b_{15}\} \) and maps it to an 18-bit codeword \( \{c_0, \ldots, c_{17}\} \). The first 5 bits of the input data \( \{b_0, \ldots, b_4\} \) are mapped directly on the first 5 bits of the codeword \( \{c_0, \ldots, c_4\} \). Bits \( \{b_5, \ldots, b_{12}\} \) of the input data are mapped directly to bits \( \{c_9, \ldots, c_{13}\} \) of the codeword.

The \( |RDS|_3 \) of these 10 bits is then calculated as \( RDS_{10} \). The remaining 6 data bits of the input data \( \{b_5, b_6, b_7, b_{13}, b_{14}, b_{15}\} \) need to be mapped onto 8 bits of the code word \( \{c_5, c_6, c_7, c_8, c_{14}, c_{15}, c_{16}, c_{17}\} \) to ensure that the \( |RDS|_3 \) of the complete codeword is 0.

Consider the 16 different sequences of 4 bits. Two of these sequences \( \{0, 0, 0, 0\} \) and \( \{1, 1, 1, 1\} \) contain no transitions while all the rest do. The presence of a transition in the deinterleaved input data to a PR4 channel causes an output of +2 or −2. The 14 sequences with transitions are divided into 3 sets on the basis of the \( |RDS|_3 \) of the sequence. These sets labeled \( A, B, C \) are enumerated in table 5.3 with sequences from set \( A \) having a \( |RDS|_3 \) of 0, set \( B \) having a \( |RDS|_3 \) of 1 and set \( C \) having a \( |RDS|_3 \) of 2.

If the codeword \( \{c_5, c_6, c_7, c_8, c_{14}, c_{15}, c_{16}, c_{17}\} \) is constructed from the concatenation of two codewords from any of the sets \( A, B, C \) then there will be guaranteed to be at least two transitions per codeword. Table 5.4 lists all the possible combinations of these sets taken two at a time and gives the \( |RDS|_3 \) of the 8 bits and the number of sequences that make up this set.

It can be seen that there are 68 sequences with \( |RDS|_3 \) of 0, 64 with \( |RDS|_3 \) of 1, and 64 with \( |RDS|_3 \) of 2. It is required to map the 6 input data bits \( \{b_5, b_6, b_7, b_{13}, b_{14}, b_{15}\} \)
\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
|RDS|_3 = 0 & |RDS|_3 = 1 & |RDS|_3 = 2 \\
\hline
1100 & 1000 & 1110 \\
1010 & 0100 & 1101 \\
0110 & 0010 & 1011 \\
1001 & 0001 & 0111 \\
0101 & & \\
0011 & & \\
\hline
\end{array}
\]

Table 5.3: Sequences of length 4 with transitions divided in sets with corresponding \(|RDS|_3\) values

| Sequence | \(|RDS|_3\) | Sets | Number of sequences |
|----------|---------------|------|---------------------|
| 0        |               | A : A | 6 \times 6 = 36     |
|          |               | B : C | 4 \times 4 = 16     |
|          |               | C : B | 4 \times 4 = 16     |
|          |               |       | Total = 68          |
| 1        |               | C : C | 4 \times 4 = 16     |
|          |               | A : B | 6 \times 4 = 24     |
|          |               | B : A | 4 \times 6 = 24     |
|          |               |       | Total = 64          |
| 2        |               | B : B | 4 \times 4 = 16     |
|          |               | A : C | 6 \times 4 = 24     |
|          |               | C : A | 4 \times 6 = 24     |
|          |               |       | Total = 64          |

Table 5.4: Combinations of the sets A, B, C with corresponding \(|RDS|_3\) and number of sequences

onto these sequences. As there are 2^6 or 64 possible input sequences, it can be seen that it is possible to perform a unique mapping for each possible \(|RDS|_3\) value.

The encoder can then be summarized as follows: The value \(RDS_{10}\) is calculated. If this is 0, one 8-bit sequence with \(|RDS|_3\) of 0 from table 5.4 is chosen using the data bits \(\{b_5, b_6, b_7, b_{13}, b_{14}, b_{15}\}\) as an index. Similarly if the value \(RDS_{10}\) is 1, one 8-bit sequences with \(|RDS|_3\) of 2 is selected. If the value \(RDS_{10}\) is 2, one 8-bit sequences with \(|RDS|_3\) of 1 is selected. In this way the total \(|RDS|_3\) of the completed codeword is 0 as desired.

The mapping required from \(\{b_5, b_6, b_7, b_{13}, b_{14}, b_{15}\}\) to \(\{c_5, c_6, c_7, c_8, c_{14}, c_{15}, c_{16}, c_{17}\}\) may be implemented using a table lookup or a logic mapping. To implement the mapping with tables requires a 64 \times 8 table for the \(RDS_{10} = 0\) case and a 64 \times 8 table for the \(RDS_{10} = 1\). The \(RDS_{10} = 2\) case can be calculated from the \(RDS_{10} = 1\) table by inverting all the output bits. In the next section an algorithmic based mapping is presented to reduce the size of tables required.
5.8.2 Algorithmic Mapping

The following method provides an algorithm to map the six data bits \(b_5, b_6, b_7, b_{13}, b_{14}, b_{15}\) to \(c_5, c_6, c_7, c_8, c_{14}, c_{15}, c_{16}, c_{17}\) where the sequence \(c_5, c_6, c_7, c_8, c_{14}, c_{15}, c_{16}, c_{17}\) is required to have a \(|RDS|_3\) of \(RDS_{req}\). This may reduce the logic required if a table mapping requires excessive silicon area.

The first 4 bits \(b_5, b_6, b_7, b_{13}\) are checked to see if they have a transition. This requires that they are not 0, 0, 0, 0 or 1, 1, 1, 1. They are then directly mapped to the bits \(c_5, c_6, c_7, c_8\). The \(|RDS|_3\) of these 4 bits are calculated and in conjunction with the required \(|RDS|_3\) value of \(RDS_{req}\), it can be determined which set \(A, B\) or \(C\) the bits \(c_{14}, c_{15}, c_{16}, c_{17}\) must come from. The remaining 2 bits \(b_{14}, b_{15}\) can then index into the appropriate set \(A, B\) or \(C\) to obtain the code bits \(c_{14}, c_{15}, c_{16}, c_{17}\). As sets \(B\) and \(C\) have 4 elements and the set \(A\) has 6 elements, the mapping is unique with 2 elements from set \(A\) left over. These 2 left over elements from set \(A\) can then be used to signify the two special cases of 0, 0, 0, 0 or 1, 1, 1, 1. This is done by using one of the left over sequences, say 1, 0, 0, 1, for the 0, 0, 0, 0 case and mapping this onto \(c_{14}, c_{15}, c_{16}, c_{17}\). The other left over sequence, say 0, 1, 1, 0 can be used for the 1, 1, 1, 1 case. The bits \(b_{14}, b_{15}\) can then index into the appropriate set \(A, B\) or \(C\) to obtain the code bits for \(c_5, c_6, c_7, c_8\). Table 5.5 shows an allocation of the sets that will implement an appropriate mapping. The sequences 0, 1, 1, 0 and 1, 0, 0, 1 can be used for the special cases.

<table>
<thead>
<tr>
<th>Index (\rightarrow)</th>
<th>Set A</th>
<th>Set B</th>
<th>Set C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (\rightarrow) 00</td>
<td>1100</td>
<td>1000</td>
<td>1110</td>
</tr>
<tr>
<td>1 (\rightarrow) 01</td>
<td>1010</td>
<td>0100</td>
<td>1101</td>
</tr>
<tr>
<td>2 (\rightarrow) 10</td>
<td>0101</td>
<td>0010</td>
<td>1011</td>
</tr>
<tr>
<td>3 (\rightarrow) 11</td>
<td>0011</td>
<td>0001</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>0110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Index mappings for the sets \(A, B\) and \(C\)

Using this method, encoding may be implemented using tables with at most 4 inputs and in some cases not all entries are required. In practice, tables with 4 inputs may be readily be implemented with combinatorial logic. Appendix F provides a formal description of the encoder in the Verilog hardware description language.

5.8.3 Channel Polarity and Codeword Inversion

To avoid problems with the channel polarity, it is required that if the data word \(C \rightarrow S\) then \(\overline{C} \rightarrow \overline{S}\). If the user input data is inverted then the sections of the user data that is
directly mapped to the codeword is automatically inverted. If the running digital sum of a sequence is \( x \) then the running digital sum of the inverted sequence is \(-x\), hence, if the RDS modulo 3 of the sequence is \( |x|_3 \) then the \( |RDS|_3 \) of the inverted sequence is \( |−x|_3 \).

It can then be seen that if a sequence has a \( |RDS|_3 \) of 0, its inverse has a \( |RDS|_3 \) of 0. If a sequence has a \( |RDS|_3 \) of 1, its inverse has a \( |RDS|_3 \) of 2 and similarly if the sequence has a \( |RDS|_3 \) of 2, its inverse has a \( |RDS|_3 \) of 1.

The mappings in table 5.5 shown previously have been assigned to facilitate the effect of codeword inversion. The mapping of indices \( i \) in the set \( A \) is assigned such that if \( A[i] → x \) then \( A[7] → x \). The mapping of indexes \( i \) in the sets \( B \) and \( C \) are assigned such that if \( B[i] → x \) then \( C[7] → x \), and, if \( C[i] → x \) then \( B[7] → x \). This accounts for the difference \( |RDS|_3 \) value required when the input word is inverted and results in the desired inversion property of the codewords.

Ideally the effect of channel polarity would be transparent to the user. This could be implemented by precoding the user data as is done for conventional PR4 channels. In effect, this causes each user data bit of logic 1 to cause a transition in the precoded data. The modulo RDS code is then applied. When the data is recovered, it is detected by the Viterbi detector and decoded from the RDS code. The precoding then detects the transitions to render the decoded data immune to the effect of channel polarity. However, this will have the effect of modifying the effect of minimum distance events. It will be shown in the decoder section that when using the code described previously, minimum distance error events will only cause errors within codeword boundaries. Precoding would allow such errors to extend across codeword boundaries and hence have implications for higher level ECC codes. It is proposed here that precoding not be employed and the polarity of the channel be detected during synchronization.

### 5.8.4 Run length constraints

Given the mapping described previously there is guaranteed to be a transition in the bits \( \{c_5, c_6, c_7, c_8\} \) and \( \{c_{14}, c_{15}, c_{16}, c_{17}\} \). When codewords are concatenated, there will be a sequence of 4 bits with a transition between every group of 5 arbitrary data bits. The worst case run length in a codeword would occur when one transition group is the sequence 1, 0, 0, 0, the 5 next bits are 0, 0, 0, 0, 0 and the next transition group is 0, 0, 0, 1. In this case the sequence may have 11 consecutive identical bits. However, for the PR4 channel, codewords are interleaved. If the interleaving is done with code word boundaries shifted by 4 positions relative to each other then the worst case run of symbols is shown in fig 5.16.

The worst case run length of identical symbols in the stream is then 14 identical symbols. The PR4 channel response is \( 1 − D^2 \) and hence, a sequence of 14 identical symbols causes 12 consecutive zero samples.
As there are at least two transitions per 18-bit codewords, there will be a minimum of 2 non-zero samples for every 18 channel symbols. This is a worst case non-zero sample density of 11.1%.

5.8.5 Correction and Decoding

In principle the decoder is the inverse mapping of the encoder. However given the encoder operations described some additional modifications may be made to improve the performance of a system using this code. These modifications allow some correction to applied to the codeword before decoding the user data.

5.8.6 Codeword Correction

The worst case run length of consecutive symbols in a given interleave is 11 and this allows at most 9 consecutive zero samples. It was shown in section 5.5 that minimum distance error events on a channel with a RDS modulo 3 code has error events of length $3n + 1$ $n = 1, 2, \ldots$. For a Dicode channel, a minimum distance error event of length $l$ requires $l - 2$ consecutive 0 channel samples for the event to be supported. For the code described the maximum run length of 0 channel samples in a particular interleave is 10 and hence the maximum length of a minimum distance error event is 12. However, the length of an error event must be 1 plus a multiple of 3 and so the only allowed minimum distance error events are those of length 4, 7, or 10.

These minimum distance error events must occur between non-zero channel samples. Hence, minimum distance error events must occur between transitions. In the encoder described, transitions are guaranteed to be located in certain areas. Fig 5.17 shows possible minimum distance error events that cross codeword boundaries. It can be seen that if a minimum distance error event crosses a boundary then it must start in the last transition group of the previous codeword. As these events can be no longer than 10, they cannot span a complete codeword.

Consider the error event denoted 'case B' in fig 5.17. This event begins in the last
transition group of codeword N and extends into the next codeword. This event must start at or after the last ±2 sample in the last transition group. The bit corresponding to this sample and all the samples after it will be in error. For sequences of set A (which have a |RDS|_3 of 0), the effect of these error events are shown in table 5.6. From this table, it can be seen that although some bits are in error (denoted 'e'), it is still possible to uniquely identify which sequence these correspond to. This is only the case when it is known that the sequence is from the set A. However, as the rest of the codeword is correct, the |RDS|_3 of the first 14 bits can be calculated and it can be determined whether the 4 bits are from the set A, B or C. The same applies to sets B and C as shown in table 5.7 and table 5.8. This codeword correction is independent of the particular mapping chosen in the encoder for mapping user bits to transition group bits. The only requirement is that the last group of transition bits occur at the end of the codeword.

![Figure 5.17: Error events that cross codeword boundaries](image)

<table>
<thead>
<tr>
<th>sequence</th>
<th>1 - D Channel Output</th>
<th>Decoded bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>x 0 -2 0</td>
<td>11ee</td>
</tr>
<tr>
<td>1010</td>
<td>x -2 +2 -2</td>
<td>101e</td>
</tr>
<tr>
<td>0101</td>
<td>x +2 -2 +2</td>
<td>010e</td>
</tr>
<tr>
<td>0011</td>
<td>x 0 +2 0</td>
<td>00ee</td>
</tr>
<tr>
<td>0110</td>
<td>x +2 0 -2</td>
<td>011e</td>
</tr>
<tr>
<td>1001</td>
<td>x -2 0 +2</td>
<td>100e</td>
</tr>
</tbody>
</table>

Table 5.6: Transition group sequences from set A with a minimum distance error event starting in the sequence

Hence, if the minimum distance error event starts in the last transition group of a codeword, then it is possible to uniquely determine what the last bits should have been and hence correct the code word.

This implies that the effect of a single minimum distance error event is to cause an error in one single codeword. Hence, at most, two adjacent user data bytes will have errors.
Table 5.7: Transition group sequences from set $B$ with a minimum distance error event starting in the sequence

<table>
<thead>
<tr>
<th>sequence</th>
<th>$1 - D$ Channel Output</th>
<th>Decoded bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>x -2 0 0</td>
<td>1ee</td>
</tr>
<tr>
<td>0100</td>
<td>x +2 -2 0</td>
<td>01ee</td>
</tr>
<tr>
<td>0010</td>
<td>x 0 +2 -2</td>
<td>001e</td>
</tr>
<tr>
<td>0001</td>
<td>x 0 0 +2</td>
<td>000e</td>
</tr>
</tbody>
</table>

Table 5.8: Transition group sequences from set $C$ with a minimum distance error event starting in the sequence

<table>
<thead>
<tr>
<th>sequence</th>
<th>$1 - D$ Channel Output</th>
<th>Decoded bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>x 0 0 -2</td>
<td>111e</td>
</tr>
<tr>
<td>1101</td>
<td>x 0 -2 +2</td>
<td>110e</td>
</tr>
<tr>
<td>1011</td>
<td>x -2 +2 0</td>
<td>101e</td>
</tr>
<tr>
<td>0111</td>
<td>x +2 0 0</td>
<td>000e</td>
</tr>
</tbody>
</table>

if the interleaving is done in a suitable manner.

This is identical to the case of a PR4 channel with a rate 8/9 code where two adjacent bytes can be corrupted by a single error event. Hence, the same high level error correcting codes (e.g., as in section 5.6) that are used with existing 8/9 line codes may be retained and the benefits of the modulo RDS code in terms of error events can be achieved.

5.8.7 Codeword Decoding

After the codeword have been corrected as described above, the user data can be decoded. Alternately both processes may be undertaken together and share information if the decoding of user data is done in an algorithmic fashion instead of a lookup table.

Given the received and corrected codeword $\{c_0, \ldots, c_{17}\}$ the user data bits $\{b_0, \ldots, b_4\}$ are $\{c_0, \ldots, c_4\}$ and the user data bits $\{b_8, \ldots, b_{12}\}$ are $\{c_9, \ldots, c_{13}\}$.

The remaining bits $\{b_5, b_6, b_7, b_{13}, b_{14}, b_{15}\}$ can be decoded from the bits $\{c_5, c_6, c_7, c_8\}$ and $\{c_{14}, c_{15}, c_{16}, c_{17}\}$. This may be done with a single lookup table of size $256 \times 6$. This table is simply the inverse of the forward mapping.

5.8.8 Algorithmic Decoding

If table mapping is too expensive in terms of cost or silicon area then the algorithm described in section 5.8.2 may be employed and the following algorithm used for decoding.

The input data is $\{c_5, c_6, c_7, c_8, c_{14}, c_{15}, c_{16}, c_{17}\}$. The bits $\{c_{14}, c_{15}, c_{16}, c_{17}\}$ are checked to see if they are the special cases of 1, 0, 0, 1 or 0, 1, 1, 0. If not, then the bits $\{b_5, b_6, b_7, b_{13}\}$
are directly mapped from \( \{c_5, c_6, c_7, c_8\} \). The bits \( \{b_{14}, b_{15}\} \) may then be decoded with combinatorial logic from \( \{c_{14}, c_{15}, c_{16}, c_{17}\} \). If the special cases are detected by considering the bits \( \{c_{14}, c_{15}, c_{16}, c_{17}\} \) then the user data \( \{b_5, b_6, b_7, b_{13}\} \) can be determined to be 0, 0, 0, 0 if \( \{c_{14}, c_{15}, c_{16}, c_{17}\} \) is 1, 0, 0, 1 or 1, 1, 1, 1 if \( \{c_{14}, c_{15}, c_{16}, c_{17}\} \) is 0, 1, 1, 0. The remaining bits \( \{b_{14}, b_{15}\} \) can then be decoded with combinatorial logic from \( \{c_5, c_6, c_7, c_8\} \).

Hence, the decoding may be performed with a reasonable logic implementation. Appendix F provides a description of the correction process and decoding process in the Verilog hardware description language.

5.8.9 Logic Implementation Complexity

Using the Verilog descriptions of the encoder, correction and decoder procedures, a gate level implementation based on the ES2 0.7um standard cell library was synthesized using the Cadence Synergy Synthesizer. Schematics of the synthesized logic are shown in appendix F. Table 5.9 tabulates the resulting complexity in terms of nand gate equivalents.

<table>
<thead>
<tr>
<th></th>
<th>Nand Gate Equivalents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>240</td>
</tr>
<tr>
<td>Correction</td>
<td>185</td>
</tr>
<tr>
<td>Decoder</td>
<td>75</td>
</tr>
</tbody>
</table>

Table 5.9: Rate 16/18 RDS mod 3 coder/decoder implementation complexity

5.8.10 Viterbi Path Memory

Based on the run length constraints and placement of transitions of the code, a bound for the path memory required can be calculated.

As there is a ±2 in the last 3 symbols of a code word due to the transitions, any minimum distance event crossing the code word boundary must start after this position. The event will have accumulated a squared distance of 4 at the codeword boundary. The event can converge within the next codeword or extend as a catastrophic sequence. If it extends as a catastrophic sequence then it will have an accumulated distance of 8 and be eliminated at the next boundary. If both paths extend through the next boundary, then an accumulated distance of 12 exists between them and one can be eliminated on this basis. This would require a path memory of 3 + 18 = 21. However, a transition must occur, at worst, at the 9th symbol of a code word. If the path memory is increased to a length of 21 + 9 = 32, then an event which has an accumulated distance of 12 after 21 symbols will have an accumulated distance of 16 after 30 symbols. In this case all non-minimum
distance error events will have an accumulated distance of 16. However, the difference in
gain between a path memory of 21 and 32 is likely to be small for random data. Fig 5.18
illustrates an example error event spanning two code words\(^3\).

5.9 Conclusion

It has been demonstrated that codes which eliminate minimum distance error events
of length 2 and 3 on the Dicode channel can improve the performance of the PR4 channel.
These DL codes provide coding gain by avoiding the effect of noise correlation in equalizing
the magnetic recording channel to the PR4 response. The increase in effective SNR is of
the order of 1.25dB.

Parity check codes were shown to achieved the desired gain but with excessive run
length constraints. Modulo RDS codes were proposed as a method of improving on the run
length constraints. These codes can be implemented with a code rate of 8/9 which is the
rate commonly used with normal PR4 or EPR4 channels. Thus, the increase in SNR may
be achieved with no rate loss. In terms of error event probabilities, the effective increase
in SNR can be of the order of 1.6dB.

Details of a sample rate 16/18 code with a RDS mod 3 of zero were presented. The
code presented guaranteed a maximum run length of 12 non-zero channel samples and a
worst case nonzero sample density of 11%. The code also uses a novel technique to confine
the effect of minimum distance error events to single codewords easing the design of higher
level error correcting codes.

---

\(^3\)This is the worst case error event in terms of path memory length. Worst case minimum distance
error events are as described previously.
Figure 5.18: Longest length path memory error event
Chapter 6

High Density Detector

6.1 Introduction

In this chapter a new equalization target and detector suitable for high density magnetic recording is presented. The target response is described and shown to be a reasonable match to the recording channel. The detector required for the target is developed and shown to allow practical implementation through the use of a number of approximations. The approximations are shown to have little impact on the detector performance. The complexity of VLSI implementation is assessed and shown to compare favorably with other advanced detector structures.

A novel method for deriving control and timing information is described. It is shown that new line codes or existing line codes can be used on the channel.

The theoretical and simulated performances of the proposed detector are compared to other existing detection systems and shown to outperform them at high recording densities.

6.2 1+D Channel with Decision Feedback

While the 1 + D channel can be used as a target on the magnetic recording channel with a DC free code[ISH95][FHOH95], this normally implies a rate loss due to the DC free constraint. Here it is proposed to use a higher rate code that is not DC free, compensating for the DC null in the magnetic recording channel through the use of decision feedback. This detector is shown to provide better performance than EPR4 at high recording densities while requiring a less computationally intensive detector. It will be denoted as an OPDDC detector (One Plus D with DC feedback).

6.3 Theory of Operation

The Lorentz recording channel at high recording densities has most of the signal power concentrated at low frequencies. This is illustrated in fig 6.1. It can be seen that the PR1
channel is quite a good match except for the presence of a DC null in the Lorentz channel. This is commonly compensated for with a DC null code with the corresponding rate loss.

![Magnitude Response](image)

Figure 6.1: Comparison of Partial Responses at a density of 3.0

However, signaling with a non-DC free code may be achieved by using decision feedback to cancel the effect of the DC null. This was proposed in [WD78] applied to a magnetic recording channel with flat equalization and feedback via an RC circuit.

It is proposed here to use decision feed back and a PR1 Viterbi detector. The purpose of feedback is to reconstruct the DC and low frequency components of the signal. The detection system is shown conceptually in fig 6.2.

![Conceptual Diagram](image)

Figure 6.2: Conceptual diagram of OPDDC detector

A number of problems exist with the description in fig 6.2. The main problem is that the decisions from the Viterbi detector are not available immediately due to the requirement of convergence in the path memory of the detector. The DC and low frequency components of
the signal that need to be reconstructed are characterized by being long in the time domain and hence would require a long feedback filter. It will be shown that these problems may be overcome.

6.3.1 Feedback Decisions

While the final decision of the Viterbi detector is not available until the path memory has converged, preliminary decisions may be used. Fig 6.3 shows the trellis for the PR1 channel. To aid the analysis later on, the states have been labeled as A and B corresponding to the last inputs being −1 (logic 0) and +1 (logic 1), respectively.

![Trellis diagram for PR1 Viterbi detector](image)

Figure 6.3: Trellis diagram for PR1 Viterbi detector

Denoting the path metrics as \( \Gamma_a \) and \( \Gamma_b \) and the branch metrics for a decoder input \( x_k \) as \( \lambda^k_a \), the update equations for the Viterbi detector at time index \( k \) can be written as

\[
\Gamma^{k+1}_a = Max(\Gamma^k_a + \lambda^{k+2}_a, \Gamma^k_b + \lambda^k_0)
\]

(6.1)

and

\[
\Gamma^{k+1}_b = Max(\Gamma^k_b + \lambda^{k+2}_b, \Gamma^k_a + \lambda^k_0)
\]

(6.2)

The feedback signal can be calculated for both path memories and the Viterbi update equations may be modified as

\[
\Gamma^{k+1}_a = Max(\Gamma^k_a + \lambda a^k_{-2}, \Gamma^k_b + \lambda b^k_0)
\]

(6.3)

and

\[
\Gamma^{k+1}_b = Max(\Gamma^k_b + \lambda b^k_{+2}, \Gamma^k_a + \lambda a^k_0)
\]

(6.4)

where \( \lambda a^k_a \) is the branch metric for an expected input of \( x \) with the feedback value from the path memory of state A taken into account. If the value fed back is \( a_k \) then the metric calculation is modified to

\[
\lambda a^k_a = -(y_k + a_k - x)^2
\]

(6.5)
6.3.2 Feedback filter

To reconstruct the DC and low frequencies a filter with a long impulse response is required. Therefore, it is proposed that an IIR filter is used. Consider the filter described by the recurrence equation

\[ y_k = y_{k-1}(1 - \alpha) + x_k\alpha \]  

(6.6)

This filter has a \( D \) transform \( H_{f0}(D) \) of

\[ H_{f0}(D) = \frac{\alpha}{1 - (1 - \alpha)D} \]  

(6.7)

In the frequency domain, \( D \rightarrow e^{-j2\pi fT_{bit}} \) and the frequency response is plotted for various values of alpha in fig 6.4. The values of alpha chosen are in the form \( \frac{1}{3} \) and may hence be implemented as a shift operation, eliminating the need for multiplies.

![Frequency response of feedback filter](image)

**Figure 6.4**: Feedback filter response with various values of \( \alpha \)

The gain of the filter at DC is 1.0. The magnitude response of the PR1 channel at DC is 2.0 and hence, a value of +2 is fed into the filter to feedback a detected logic 1 and a value of -2 is fed into the filter to feedback a detected logic 0.

6.3.3 Target Response

The basic response of the detector is the PR1 response of \( 1 + D \). However, the equalizer is required to equalize to a target of the \( 1 + D \) minus the signal fed back. Let the channel response be \( H_{\text{chan}}(f) \). The input sample is \( x_k \), the detected symbols in the path memories will be \( \ldots, \hat{x}_{k-3}, \hat{x}_{k-2}, \hat{x}_{k-1} \). If the previously detected symbols are correct then the desired
signal into the detector is

\[ T_{\text{Target}}(D) = 1 + D - 2DH_f(D) \]  \hspace{1cm} (6.8)

Fig 6.5 shows the detector target response with \( \alpha = 0.125 \) and the magnetic recording channel response at a density of 3.0. It can be seen that the target response models the channel reasonably well.

![Figure 6.5: Target response \( \alpha = 0.125 \) and channel response at density of 3.0](image)

Fig 6.6 shows the equalizer magnitude response for the OPDDC detector with \( \alpha = 0.125 \) and the corresponding EPR4 equalizer. The better channel matching of the OPDDC target response results in an equalizer with less high frequency boosting than the EPR4 response. However, the equalizer for the OPDDC target has a required phase response as shown in fig 6.7. This phase response is almost a 90 degree phase shift over the frequencies of interest.

### 6.4 Detector Update Equations

The updating of the path metrics requires the calculation of alternative branch metrics. The update equations can be written

\[ \Gamma_a^{k+1} = \max(\Gamma_a^k + \lambda a_{-2}^k, \Gamma_b^k + \lambda b_0^k) \]
\[ \Gamma_b^{k+1} = \max(\Gamma_b^k + \lambda b_{+2}^k, \Gamma_a^k + \lambda a_0^k) \]  \hspace{1cm} (6.9)

with \( \lambda a_{-2}^k = -(y_k + a_k - x)^2 \) and \( \lambda b_{+2}^k = -(y_k + b_k - x)^2 \) where \( a_k \) and \( b_k \) are the values fed back through the feedback filter from the path memories \( A \) and \( B \) respectively.

As there are only two path metrics, and only the difference between the path metric values is important, the difference between the two metrics may be updated and stored as
in the case of detectors for the Dicode channel[CDH+92]. The update equations may be written as a single update of

\[
\Gamma_a^{k+1} - \Gamma_b^{k+1} = \text{Max}(\Gamma_a^k + \lambda a_{k-2}^k, \Gamma_b^k + \lambda b_0^k) - \text{Max}(\Gamma_a^k + \lambda b_{k+2}^k, \Gamma_b^k + \lambda a_0^k)
\]

\[
= \text{Max}(\Gamma_a^k - (y_k + a_k + 2)^2, \Gamma_b^k - (y_k + b_k)^2)
- \text{Max}(\Gamma_b^k - (y_k + b_k - 2)^2, \Gamma_a^k - (y_k + a_k)^2)
\]

\[
= \text{Max}(\Gamma_a^k - y_k^2 - 2a_ky_k - 4y_k - 4a_k - a_k^2 - 4, \Gamma_b^k - y_k^2 - 2b_ky_k - b_k^2)
- \text{Max}(\Gamma_b^k - y_k^2 - 2b_ky_k + 4y_k + 4b_k - b_k^2 - 4, \Gamma_a^k - y_k^2 - 2a_ky_k - a_k^2)
\]

\[(6.10)\]
Noting that $Max(x + z, y) = z + Max(x, y - z)$, common terms may be canceled and the update equation may be written as
\[
\frac{\Gamma_{a}^{k+1} - \Gamma_{b}^{k+1}}{4} = Max\left(\frac{\Gamma_{a}^{k} - \Gamma_{b}^{k}}{4}, y_{k} + a_{k} + 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})\right) - y_{k} - a_{k} - 1
\]
\[\quad - Max\left(y_{k} + b_{k} - 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})\Gamma_{a}^{k} - \Gamma_{b}^{k}\Gamma_{b}^{k}\right)\]  

(6.11)

Denoting the difference metric as $\Delta^{k} = \frac{\Gamma_{a}^{k} - \Gamma_{b}^{k}}{4}$ the update equation becomes
\[
\Delta^{k+1} = Max\left(\Delta^{k}, y_{k} + a_{k} + 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})\right) - y_{k} - a_{k} - 1
\]
\[\quad - Max\left(y_{k} + b_{k} - 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})\Delta^{k}\right)\]  

(6.12)

6.4.1 Path Memory Updates

Let the elements of the path memory for state $A$ at the time index $k$ be denoted $[PA]^{k} = \{\ldots, p_{a_{k-1}}, p_{a_{k}}\}$ and the elements of the path memory for state $B$ be denoted $[PB]^{k} = \{\ldots, p_{b_{k-1}}, p_{b_{k}}\}$. The path memories can be updated using the register exchange method [CDH+92]. This involves either shifting the path memory or loading the path memory with the contents of a different path memory. The new data bit for the path is then placed at the end of the path memory. Loading the path memory and appending the data bit $b$ will be denoted $[PX]^{k+1} = [PY]^{k}b$ were $[PX]$ is the path to be updated. By considering the trellis for the $1 + D$ channel response as in fig 6.3 and the update equations in eqn 6.12, the required path memory updates can be derived.

When the term $\Delta^{k}$ is greater than the term $y_{k} + a_{k} + 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})$ in the first maximum operator of eqn 6.12, the path memory for state $A$ is updated as
\[
[PA]^{k+1} = [PA]^{k}||0
\]

(6.13)

otherwise
\[
[PA]^{k+1} = [PB]^{k}||0
\]

(6.14)

Similarly, if $y_{k} + b_{k} - 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})$ is greater than $\Delta^{k}$ in the second maximum operator of eqn 6.12, then the path memory for state $B$ is updated as
\[
[PB]^{k+1} = [PB]^{k}||1
\]

(6.15)

otherwise
\[
[PB]^{k+1} = [PA]^{k}||1
\]

(6.16)

However, not all possible combinations of these updates may occur. Consider, the case of $\Delta^{k} > y_{k} + a_{k} + 1 + \frac{(a_{k} - b_{k})}{2}(y_{k} + \frac{a_{k} + b_{k}}{2})$. The update for state $A$ is
\[
[PA]^{k+1} = [PA]^{k}||0
\]

(6.17)
But
\[ y_k + b_k - 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) \]
\[ = y_k + a_k + 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) + b_k - 1 - a_k - 1 \]
\[ < \Delta^k + b_k - a_k - 2 \] (6.18)

However, \( a_k \) is the result of filtering the path memory \([PA]^k\) through the filter \( H_{fb}(D)\) and \( b_k \) is the result of filtering the path memory \([PB]^k\) through the filter \( H_{fb}(D)\) and it will be seen later (section 6.5) that this provides a bound on \( a_k - b_k \) of
\[ 0 \geq a_k - b_k \geq -4\alpha \] (6.19)
and hence
\[ y_k + b_k - 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) < \Delta^k + 4\alpha - 2 \] (6.20)
This implies that the update for the path \([PB]^k\) must be \([PB]^{k+1} = [PA]^k||1\), provided \( \alpha \leq \frac{1}{2} \).

Similarly if \( y_k + b_k - 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) > \Delta^k \) then the path \([PB]^k\) is updated as \([PB]^{k+1} = [PB]^k||1\).

Now
\[ y_k + a_k + 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) \]
\[ = y_k + b_k - 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) + a_k + 1 - b_k + 1 \]
\[ > \Delta^k + 2 + a_k - b_k \] (6.21)

With \( 0 \geq a_k - b_k \geq -4\alpha \) and \( \alpha \leq \frac{1}{2} \), then
\[ y_k + a_k + 1 + \frac{(a_k - b_k)}{2} (y_k + \frac{a_k + b_k}{2}) > \Delta^k \] (6.22)

Therefore, if the update for state \( B \) is \([PB]^{k+1} = [PB]^k||1\) then the update for state \( A \) must be \([PA]^{k+1} = [PB]^k||0\).

Fig 6.8 shows graphically the allowed updates. The paths can either swap over or converge. However, both paths may not extend simultaneously.

![Figure 6.8: Allowable path memory updates](image-url)
6.4.2 Feedback Update

The feedback signals $a_k$ and $b_k$ are in principle generated by filtering the data in the path memories with an IIR filter. However, the data in the path memories may change completely when a path memory converge occurs. This is handled by updating the feedback signals based on the path memory operations.

The feedback signal from the path memory for state $A$ at time $k$ is denoted $a_k$ and the feedback signal from the path memory for state $B$ at time $k$ is denoted $b_k$. At the time index $k+1$ both $a_{k+1}$ and $b_{k+1}$ are required. These need to be calculated using the difference equation in eqn 6.6. The value required for $a_k$ is

$$a_{k+1} = \Phi(1 - \alpha) + 2\alpha p_{a_{k+1}}$$  \hspace{1cm} (6.23)

where $\Phi$ is the filtered output of the path memory values \{...,$p_{a_{k-1}},p_{a_k}$\}. However, $p_{a_{k+1}}$ will always be a logic 0 corresponding to a value of $-1$ as in eqns 6.13 and 6.14. The rest of the path memory \{...,$p_{b_{k-1}},p_{b_k}$\} will be unchanged if there was no path convergence or will be \{...,$p_{b_{k-1}},p_{b_k}$\} if there was a convergence in the path memory. Thus, the value of $\Phi$ due to the path memory of state $A$ will be $a_k$ if there was no convergence and will be $b_k$ if there was a convergence. Similarly the value of $b_{k+1}$ can be updated. Table 6.1 tabulates the update operations for the feedback terms. Hence, the IIR feedback may be

<table>
<thead>
<tr>
<th>Path Memory Update</th>
<th>$a_{k+1} =$</th>
<th>$b_{k+1} =$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[PA]^{k+1} = [PA]^k [0]$</td>
<td>$a_k(1 - \alpha) - 2\alpha$</td>
<td>$a_k(1 - \alpha) + 2\alpha$</td>
</tr>
<tr>
<td>$[PA]^{k+1} = [PB]^k [0]$</td>
<td>$b_k(1 - \alpha) - 2\alpha$</td>
<td>$b_k(1 - \alpha) + 2\alpha$</td>
</tr>
<tr>
<td>$[PA]^{k+1} = [PB]^k [0]$</td>
<td>$b_k(1 - \alpha) - 2\alpha$</td>
<td>$b_k(1 - \alpha) + 2\alpha$</td>
</tr>
</tbody>
</table>

Table 6.1: Updates for feedback terms

readily implemented.

6.5 Practical Consideration for Detector

The update equation in eqn 6.12 involves the term $\frac{(a_k + b_k)}{2}(y_k + \frac{a_k + b_k}{2})$ which would require a multiplier to implement exactly. However, this term may be approximated as follows. By inspecting table 6.1 it can be seen that when a path convergence occurs the difference between $a_{k+1}$ and $b_{k+1}$ is

$$a_{k+1} - b_{k+1} = [a_k(1 - \alpha) - 2\alpha] - [a_k(1 - \alpha) + 2\alpha] = -4\alpha$$  \hspace{1cm} (6.24)
\[ a_{k+1} - b_{k+1} = \left[b_k(1 - \alpha) - 2\alpha\right] - \left[b_k(1 - \alpha) + 2\alpha\right] = -4\alpha \]  

(6.25)

which in both cases is \(-4\alpha\). The only other possibility is that the path memories swap over. Consider a path convergence followed by a sequence of swaps. After the path convergence let \(a_{k+1} = C - 2\alpha\) and \(b_{k+1} = C + 2\alpha\) where \(C\) is a constant and the difference \(a_{k+1} - b_{k+1} = -4\alpha\). When the path memories swap at index \(k + 2\) the new difference will be

\[ a_{k+2} - b_{k+2} = \left[b_{k+1}(1 - \alpha) - 2\alpha\right] - \left[a_{k+1}(1 - \alpha) + 2\alpha\right] = -(a_{k+1} - b_{k+1})(1 - \alpha) - 4\alpha \]  

(6.26)

The difference at index \(k + 2\) is the difference at index \(k + 1\) multiplied by the factor \(-(1 - \alpha)\) and \(4\alpha\) subtracted. The initial difference after a convergence is \(-4\alpha\). If the factor \(1 - \alpha\) is approximated as 1, then the difference follows the sequence \(-4\alpha, 0, -4\alpha, 0, -4\alpha, \ldots\) The factor \(1 - \alpha\) causes the difference to converge to a value between 0 and \(-4\alpha\). It can also be seen that \(a_k - b_k\) is bounded by

\[ 0 \geq a_k - b_k \geq -4\alpha \]  

(6.27)

It is proposed to approximate the term \(\frac{(a_k - b_k)}{2}\) by 0 or \(-2\alpha\). When a path memory convergence has occured the value of \(-2\alpha\) is used and is the correct value. When a path memory swap occurs a value of 0 is used. If a subsequent swap occurs, \(-2\alpha\) is used. Thus, a sequence of swaps causes an alternating pattern to be used as an approximation for the correct value. By inspection of the trellis in fig 6.3, it can be seen that a path memory swap occurs when the channel sample is of 0 value. Fig 6.9 shows the error involved with the approximation as a function of run lengths of zero.

However, any suitable line code will seek to minimize the run length of 0 channel samples to ensure sufficient information for AGC and timing recovery control loops. Hence, the approximation error should be quite small. By using this approximation, the multiplication by \(\frac{(a_k - b_k)}{2}\) is replaced by multiplication by 0 or \(-2\alpha\). As \(\alpha\) is a negative power of 2, the multiplication may be replaced by a shift operation.

Fig 6.10 shows the simulated effective SNR of the detector with and without this approximation. The simulations are based on recording densities of 2.5, 3.0 and 3.5 bits per \(PW_{50}\) with \(\alpha\) being 0.25, 0.125 and 0.0625 respectively. It can be seen that there is no significant loss due to this approximation. The analysis in section 6.8 confirms that this is the case.
6.5.1 Update Equations

The update equations for the difference metric were given in eqn 6.12.

\[
\Delta^{k+1} = \text{Max} \left( \Delta^k, y_k + a_k + 1 + \frac{(a_k - b_k)}{2}(y_k + \frac{a_k + b_k}{2}) \right) - y_k - a_k - 1
\]

\[
-\text{Max} \left( y_k + b_k + 1 + \frac{(a_k - b_k)}{2}(y_k + \frac{a_k + b_k}{2}), \Delta^k \right)
\]  

(6.28)

The term \( \frac{(a_k - b_k)}{2} \) will be approximated as described previously and the approximation value is denoted \( \delta_k \) with \( \delta_k \in \{0, -2\alpha\} \).

At the beginning of an update the following information is available:

\[
\delta_k, a_k, b_k, \Delta^k \text{ and } y_k
\]

(6.29)
The term \(\frac{(a_{k+1}+b_{k+1})}{2}\) can be calculated with an addition and a shift operation. The term 
\(y_k + a_k + 1 + \frac{(a_k-b_k)}{2}(y_k + \frac{a_k+b_k}{2})\) may be approximated as
\[L_{a_k} = y_k + a_k + 1 + \delta_k(y_k + \frac{a_k+b_k}{2})\] (6.30)
and the term 
\(y_k + b_k - 1 + \frac{(a_k-b_k)}{2}(y_k + \frac{a_k+b_k}{2})\) may be approximated as
\[L_{b_k} = y_k + b_k - 1 + \delta_k(y_k + \frac{a_k+b_k}{2})\] (6.31)

The difference metric may then be calculated as
\[\Delta^{k+1} = \text{Max} (\Delta^k, L_{a_k}) - y_k - a_k - 1 - \text{Max} (L_{b_k}, \Delta^k)\] (6.32)

Define two binary variables \(sa_k\) and \(sb_k\) to represent the results of the maximum operators
\[
sa_k = \begin{cases} 
1 & \Delta^k > L_{a_k} \\ 
0 & \text{otherwise} 
\end{cases}
\] (6.33)
\[
sb_k = \begin{cases} 
1 & L_{b_k} > \Delta^k \\ 
0 & \text{otherwise} 
\end{cases}
\] (6.34)

The approximation \(\delta_{k+1}\) may be updated as in the table 6.2. The feedback terms \(a_{k+1}\) and \(b_{k+1}\) and path memories may be updated based on \(sa_k\) and \(sb_k\) as in table 6.3.

<table>
<thead>
<tr>
<th>(sa_k)</th>
<th>(sb_k)</th>
<th>(\delta_k)</th>
<th>(\delta_{k+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2(\alpha)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2(\alpha)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2(\alpha)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-2(\alpha)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.2: Updates for \(\delta_{k+1}\)

<table>
<thead>
<tr>
<th>(sa_k)</th>
<th>(sb_k)</th>
<th>(a_{k+1})</th>
<th>(b_{k+1})</th>
<th>(PA^{k+1})</th>
<th>(PB^{k+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>(b_k(1-\alpha) - 2\alpha)</td>
<td>(a_k(1-\alpha) + 2\alpha)</td>
<td>(PA^{k})</td>
<td>(PB^{k})</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(a_k(1-\alpha) - 2\alpha)</td>
<td>(a_k(1-\alpha) + 2\alpha)</td>
<td>(PA^{k})</td>
<td>(PB^{k})</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>(b_k(1-\alpha) - 2\alpha)</td>
<td>(b_k(1-\alpha) + 2\alpha)</td>
<td>(PA^{k})</td>
<td>(PB^{k})</td>
</tr>
</tbody>
</table>

Table 6.3: Updates for feedback terms

Hence, at the end of the update the following updated information is available.
\[
\delta_{k+1} \quad a_{k+1} \quad b_{k+1} \quad \Delta^{k+1}
\] (6.35)

and the next update will occur when \(y_{k+1}\) arrives. This algorithm can be described sequentially as in table 6.4.
INITIAL: Initialize all variables to 0

REPEAT

BEGIN OPDDC DETECTOR

\[ L_{a_k} = a_k + 1 + \left[ y_k + \delta_k (y_k + \frac{a_k + b_k}{2}) \right] \]
\[ L_{b_k} = b_k - 1 + \left[ y_k + \delta_k (y_k + \frac{a_k + b_k}{2}) \right] \]

IF \( \delta_k = -2\alpha \) THEN \( \delta_{k+1} = 0 \) ELSE \( \delta_{k+1} = -2\alpha \)

IF \( \Delta^k > L_{a_k} \) THEN

\[ \Delta^{k+1} = \Delta^k \]
\[ [PA]^{k+1} = [PA]^k \parallel 0 \]
\[ a_{k+1} = a_k (1 - \alpha) - 2\alpha \]
\[ b_{k+1} = -2\alpha \]

ELSE

\[ \Delta^{k+1} = L_{a_k} \]
\[ [PA]^{k+1} = [PB]^k \parallel 0 \]
\[ a_{k+1} = b_k (1 - \alpha) - 2\alpha \]

END IF

IF \( L_{b_k} > \Delta^k \) THEN

\[ \Delta^{k+1} = \Delta^{k+1} - L_{b_k} \]
\[ [PB]^{k+1} = [PB]^k \parallel 1 \]
\[ b_{k+1} = b_k (1 - \alpha) + 2\alpha \]
\[ \delta_{k+1} = -2\alpha \]

ELSE

\[ \Delta^{k+1} = \Delta^{k+1} - \Delta^k \]
\[ [PB]^{k+1} = [PA]^k \parallel 1 \]
\[ b_{k+1} = a_k (1 - \alpha) + 2\alpha \]

END IF

\[ \Delta^{k+1} = \Delta^{k+1} - y_k - a_k - 1 \]

END OPDDC DETECTOR

Table 6.4: OPDDC detection algorithm
All this processing may be achieved with addition and shifting operations. Fig 6.11 shows a block diagram illustrating the operations required to implement the detector directly as described. The block diagram includes the feedback updates. The block diagram contains the equivalent of 15 two-operand adders. However, many of the adders add a term multiplied by $\alpha$ (shift down a number of bits) to another operand and hence do not require a complete adder. A direct implementation of an EPR4 detector would require 8 states with 3 full adders per state or 24 adders.

Figure 6.11: Direct implementation of OPDDC detector

6.5.2 Detector Pipelining

While the update equations were described in a serial fashion, practical high speed implementations of the detection algorithm requires appropriate pipelining of the required calculations. The basic operation of the detector is the normal Viterbi detector operation of computing branch metrics, adding these to existing path metrics and selecting the maximum new path metrics. In section 6.5.1 the values for $La_{k+1}$ and $Lb_{k+1}$ would be calculated based on the values of $\delta_{k+1}$, $a_{k+1}$, $b_{k+1}$ and $y_{k+1}$. However, it is possible to compute all the possibilities of $La_{k+1}$ and $Lb_{k+1}$ based on $a_k$, $b_k$ and $y_{k+1}$ and select the correct one when $sa_k$, $sb_k$ and $\delta_k$ become available.

\[1\] the value $y_{k+1}$ is effectively available by notionally delaying all other variables
Consider \( L_{a_{k+1}} = y_{k+1} + a_{k+1} + 1 + \delta_{k+1}(y_{k+1} + \frac{a_{k+1} + b_{k+1}}{2}) \). The possible values for \( L_{a_{k+1}} \) will depend on \( sa_k \) and \( sb_k \) and \( \delta_k \).

If \( sa_k = 0 \) and \( sb_k = 0 \) and \( \delta_k = 0 \) then

\[
L_{a_{k+1}} = y_{k+1} + a_{k+1} + 1 + \delta_{k+1}(y_{k+1} + \frac{a_{k+1} + b_{k+1}}{2})
\]

\[
= y_{k+1} + b_k(1 - \alpha) - 2\alpha + 1 - 2\alpha \left( y_{k+1} + \frac{b_k(1 - \alpha) - 2\alpha + a_k(1 - \alpha) + 2\alpha}{2} \right)
\]

\[
= y_{k+1} + b_k(1 - \alpha) - 2\alpha + 1 - 2\alpha y_{k+1} - ab_k(1 - \alpha) - aa_k(1 - \alpha)
\]

\[
= y_{k+1} - 2\alpha + 1 - 2\alpha y_{k+1} + b_k(1 - \alpha) - ab_k(1 - \alpha) - aa_k(1 - \alpha)
\]

(6.36)

This can be repeated for the 4 possible cases and the results are summarized in table 6.5 and table 6.6.

<table>
<thead>
<tr>
<th>( sa_k )</th>
<th>( sb_k )</th>
<th>( \delta_k )</th>
<th>( L_{a_{k+1}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( y_{k+1} - 2\alpha + 1 - 2\alpha y_{k+1} + b_k(1 - \alpha) - ab_k(1 - \alpha) - aa_k(1 - \alpha) )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2\alpha</td>
<td>( y_{k+1} - 2\alpha + 1 + b_k(1 - \alpha) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( y_{k+1} - 2\alpha + 1 - 2\alpha y_{k+1} + b_k(1 - \alpha) - ab_k(1 - \alpha) - aa_k(1 - \alpha) )</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( y_{k+1} - 2\alpha + 1 - 2\alpha y_{k+1} + b_k(1 - \alpha) - ab_k(1 - \alpha) - aa_k(1 - \alpha) )</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: All possible values for \( L_{a_{k+1}} \)

<table>
<thead>
<tr>
<th>( sa_k )</th>
<th>( sb_k )</th>
<th>( \delta_k )</th>
<th>( L_{b_{k+1}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( y_{k+1} + 2\alpha - 1 - 2\alpha y_{k+1} + a_k(1 - \alpha) - ab_k(1 - \alpha) - aa_k(1 - \alpha) )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2\alpha</td>
<td>( y_{k+1} + 2\alpha - 1 + a_k(1 - \alpha) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( y_{k+1} + 2\alpha - 1 - 2\alpha y_{k+1} + a_k(1 - \alpha) - 2aa_k(1 - \alpha) )</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( y_{k+1} + 2\alpha - 1 - 2\alpha y_{k+1} + b_k(1 - \alpha) - 2ab_k(1 - \alpha) )</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.6: All possible values for \( L_{b_{k+1}} \)

In these tables the terms may be split into terms that depend on \( a_k \) and \( b_k \) and terms that do not. For example the term \( y_{k+1} - 2\alpha + 1 - 2\alpha y_{k+1} \) may be computed outside the main Viterbi feedback loop and does not contribute to limiting the 'ACS' bottleneck. Table 6.7 lists all of these terms.

\[
\begin{align*}
y_{k+1} - 2\alpha + 1 - 2\alpha y_{k+1} \\
y_{k+1} - 2\alpha + 1 \\
y_{k+1} + 2\alpha - 1 - 2\alpha y_{k+1} \\
y_{k+1} + 2\alpha - 1
\end{align*}
\]

Table 6.7: Terms independent of \( a_k \) and \( b_k \)

All other terms in table 6.5 and table 6.6 may be calculated with the terms \( a_k(1 - \alpha) \)
and $b_k(1 - \alpha)$ and hence, it would be advantageous to update and store these terms instead of calculating them. These are updated as in table 6.8.

<table>
<thead>
<tr>
<th>$sa_k$</th>
<th>$a_{k+1}(1 - \alpha)$</th>
<th>$sb_k$</th>
<th>$b_{k+1}(1 - \alpha)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$(b_k(1 - \alpha)) - \alpha(a_k(1 - \alpha)) - 2\alpha + 2\alpha^2$</td>
<td>0</td>
<td>$(a_k(1 - \alpha)) - \alpha(a_k(1 - \alpha)) + 2\alpha - 2\alpha^2$</td>
</tr>
<tr>
<td>1</td>
<td>$(a_k(1 - \alpha)) - \alpha(a_k(1 - \alpha)) - 2\alpha + 2\alpha^2$</td>
<td>1</td>
<td>$(b_k(1 - \alpha)) - \alpha(b_k(1 - \alpha)) + 2\alpha - 2\alpha^2$</td>
</tr>
</tbody>
</table>

Table 6.8: Updates for $a_{k+1}(1 - \alpha)$ and $b_{k+1}(1 - \alpha)$

In this way, an update starts with $La_{k+1}$, $Lb_{k+1}$ and $\Delta^{k+1}$ known. The metric comparisons may proceed immediately. While the comparisons are being computed, it is possible to simultaneously calculate the 3 possible new values of $\Delta^{k+2}$ as listed in table 6.9 and select the appropriate one when the comparison is complete. Fig 6.12 shows the pipeline operation. With this pipeline, the maximum speed of the detector would be determined by the delay of a 4-operand adder and a multiplexor or the comparison operation and select. The requirement for multiple input addition operations suggests that carry save arithmetic might provide some speed advantages[Omo94] as used in multiplier designs. The amount of pipelining required in a practical design would depend on the required speed of operation and where the critical path occurs. Detailed design targeted towards a particular IC process would be required to identify the critical path.

$$
\begin{align*}
\Delta^{k+1} - \Delta^{k+1} - y_{k+1} - a_{k+1} - 1 &= -y_{k+1} - a_{k+1} - 1

\Delta^{k+1} - \Delta^{k+1} - y_{k+1} - a_{k+1} - 1 &= -y_{k+1} - a_{k+1} - 1
\end{align*}
$$

Table 6.9: Possible updates for $\Delta^{k+2}$

Even higher speeds might be achieved by operating on two samples per cycle. As the detector has only two states, this should be more practical than using two samples per cycle in an EPR4 detector which has 8 states.

The ability to pipeline the detector in this way provides a tradeoff between speed of operation and implementation complexity.

### 6.6 Control Loops

In any data channel, provision must be made for various control loops. In PRML channels, this typically includes gain, timing and equalization adaptation. In the existing PR4 type channels this is usually achieved by making preliminary decisions based on threshold detection of the data samples and using these decisions with gradient based update algorithms [CDH+92]. This has the advantage that acquisition may achieved without
the detector delay. Excessive delay in these feedback loops can lead to instability or long acquisition periods.

With the proposed detector target, there is not a simple threshold detection scheme to make preliminary data decisions. Here, a new method is proposed to avoid this problem.

The target response for the channel is

$$T(D) = 1 + D - 2D H_{fb}(D)$$  \hspace{1cm} (6.37)

The feedback filter has a $D$ transform of

$$H_{fb}(D) = \frac{\alpha}{1 - (1 - \alpha)D}$$  \hspace{1cm} (6.38)

Hence

$$T(D) = 1 + D - \frac{2\alpha D}{1 - (1 - \alpha)D}$$  \hspace{1cm} (6.39)
Consider taking a signal $W(D)$ corresponding to a sequence $w_k$ and negating every second sample of the sequence. This modulated sequence $W'(D)$ will have a $D$ transform of

$$W'(D) = \sum_{k=0}^{\infty} w_k(-1)^k D^k$$

$$= \sum_{k=0}^{\infty} w_k(-D)^k$$

$$= W(-D) \quad (6.40)$$

Let the signal $W(D)$ be a data signal $X(D)$ passed through the target response $T(D)$

$$W(D) = X(D)T(D) \quad (6.41)$$

This signal is then modulated as described to get $W'(D)$

$$W'(D) = X(-D)T(-D) \quad (6.42)$$

and passed through a filter with the target response to give a signal $Y'(D)$ with

$$Y'(D) = X(-D)T(-D)T(D) \quad (6.43)$$

This signal is then modulated again resulting in a signal $Y(D)$ with

$$Y(D) = X(D)T(D)T(-D) \quad (6.44)$$

If this operation is performed with $T(D) = 1 + D - \frac{2\alpha D}{1 - (1-\alpha)D}$, then the resulting output signal is

$$Y(D) = X(D) \left(1 + D - \frac{2\alpha D}{1 - (1-\alpha)D}\right) \left(1 - D + \frac{2\alpha D}{1 + (1-\alpha)D}\right)$$

$$= X(D) \left(1 - D + \frac{2\alpha D}{1 + (1-\alpha)D} + D - D^2 + \frac{2\alpha D^2}{1 + (1-\alpha)D}ight.$$

$$- \frac{2\alpha D}{1 - (1-\alpha)D} + \frac{2\alpha D^2}{1 - (1-\alpha)D} - \frac{4\alpha^2 D^2}{(1 - (1-\alpha)D)(1 + (1-\alpha)D)}$$

$$\ldots$$

$$= X(D)(1 - D^2) \quad (6.45)$$

which is the input signal passed through a PR4 channel response of $1 - D^2$. This allows samples that are equalized to the target response for the OPDCC channel to be simply converted to PR4 samples. Fig 6.13 shows the operations required to achieved this filtering operation. With $\alpha$ chosen as a negative power of 2, this operation can be readily performed with shift and addition operations.

With PR4 samples readily obtainable from the OPDCC samples, the existing algorithms for gain, timing recovery and LMS filter adaption used in PR4 data channels may be used without modification. The block diagram in fig 6.14 shows the proposed system.
Figure 6.13: Operations to recover PR4 samples from OPDDC equalized samples

Figure 6.14: Block diagram of AGC, timing recovery and LMS system

6.7 Line Coding

For the OPDDC channel the detector is based on the PR1 polynomial of $1 + D$ after the decision feedback is applied. The path memories converge when a PR1 channel sample has a value of ±2.

In order to render the detected data immune to the polarity of the channel, precoding can be used.

The PR1 channel output is

$$y_k = x_k + x_{k-1}$$  \hspace{1cm} (6.46)

Define the precoded data $b_k$ in terms of the user data $a_k$ as the logic mapping

$$b_k = a_k \oplus b_{k-1}$$  \hspace{1cm} (6.47)
with $\oplus$ representing the exclusive-or function. If the mapping of the logic from $1 \to +1$ and $0 \to -1$ is applied the precoding becomes

$$b_k = a_k b_{k-1} \quad (6.48)$$

the channel input is $b_k$ and hence the channel output is

$$y_k = b_k + b_{k-1}$$

$$= a_k b_{k-1} + b_{k-1}$$

$$= (a_k + 1)b_{k-1} \quad (6.49)$$

As $b_k \in \{-1, +1\}$

$$y_k = \begin{cases} 
\pm 2 & a_k = 1 \\
0 & a_k = -1 
\end{cases} \quad (6.50)$$

Hence, the channel symbol is $\pm 2$ for a logic 1 user bit to the precoder or 0 for a logic 0 bit into the precoder.

To limit the run length of zero channel samples, the run length of logic 0’s into the precoder must be constrained. If the control and timing loops are based on the PR4 samples as in section 6.6 then the run lengths of $+2$ and the run lengths of $-2$ must be also be constrained as these must be filtered by $1 - D$ to give PR4 samples and hence, run lengths of identical samples need to be constrained. A run length of $L$ identical samples from the PR1 channel causes $L - 1$ zeros from the PR4 channel. A suitable code constraint can then be designed, as described in the next section.

### 6.7.1 Sample Code Constraint

Let the desired maximum run length of zero samples through a PR4 channel for control loop purposes be 5 and the maximum run length of PR1 zero samples to limit the Viterbi path memory be 6. This requires the data into the precoder has the constraints

1. The maximum run of 0’s is 6 to meet the path memory requirement of the PR1 channel.
2. The maximum run of 1’s is $5 + 1 = 6$ to meet the gain and timing requirements through the PR4 channel.
3. The maximum run of 0’s is $5 + 1 = 6$ to meet the gain and timing requirements through the PR4 channel.

Fig 6.15 shows a connected graph illustrating this constraint. The capacity of the constraint is 0.988109 and the techniques describe in [MSW92] may be used to design a finite state encoder with state independent decoder with any rate less than or equal to this capacity.
6.7.2 Existing Line Codes

Consider the precoder used on existing PR4 channels of

\[ b_k = a_k \oplus b_{k-2} \]  \hspace{1cm} (6.51)

for precoding and

\[ a_k = b_k \oplus b_{k-2} \]  \hspace{1cm} (6.52)

for recovering the original data.

In order to use a finite path memory for the detector, the run length of zero samples through a PR1 needs to be constrained. The logic sequence \( b_k = \{\ldots, 1, 0, 1, 0, 1, 0, \ldots\} \) is required on the channel to produce continuous zero samples when converted to bipolar values and filtered with the \( 1+D \) channel response. This requires the data into the precoder to be \( a_k = \{\ldots, 0, 0, 0, 0, 0, \ldots\} \). Similarly, the sequence of \( b_k = \{\ldots, 1, 1, 1, 1, 1, 1, \ldots\} \) and \( b_k = \{\ldots, 0, 0, 0, 0, 0, 0, \ldots\} \) results from the input sequence \( a_k = \{\ldots, 0, 0, 0, 0, 0, \ldots\} \) (but with different precoder initial states). These sequences cause runs of +2 channel samples and runs of −2 channel samples respectively. Run lengths of identical symbols can hence be eliminated by using a line code that limits the run length of logic 0’s into the precoder. This is readily achieved by using the existing line codes for PRML channels and existing precoding.

Consider an existing rate 8/9 \((0,4/4)\) code as used on PRML channels [Wol91]. This limits the number of consecutive logic 0’s to the precoder to be 4 (the global ‘k’ constraint) and hence limits the run length of zero samples on the PR1 channel. This serves to limit the path memory of the OPDDC Viterbi detector.

If the gain, timing and adaption loops are based on PR4 samples as described in
section 6.6, then these loops will receive sufficient non-zero samples for stable loops as in existing PRML control loops.

6.8 Theoretical Performance

The basic core of the detector is a PR1 Viterbi detector. The performance of such a detector is the presence of correlated noise is derived in section 3.9.1. It was shown there that the minimum distance error events of length \( l \) are of the form

\[
e_k = \begin{cases} 
\pm\{2,0,\ldots,0,2\} & \text{if } l \text{ even} \\
\pm\{2,0,\ldots,0,-2\} & \text{if } l \text{ odd}
\end{cases}
\]  

(6.53)

Consider the minimum distance error event shown in fig 6.16.

Assume that the correct path is chosen up to time index \( k - 1 \) and then an error event occurs. Assume the correct path corresponded to the \( \ldots,-2,-2,\ldots \) sequence but that the path \( \ldots,0,0,\ldots \) is chosen in error. This corresponds to a minimum distance error event on the PR1 channel.

In the case of the decision feedback in the proposed detector, the distance between the two paths will be modified. At time index \( k - 1 \), both the feedback signal for the \(-2\) and \(0\) extension from state A will be the same and will be the correct value as all decisions up to that point are assumed to be correct. Hence, \( e_{k-1} = -2 - 0 \). However, at time index \( k \), this is not the case. The correct path extension of \(-2\) from state A will have the correct feedback signal. However, the path memories at index \( k - 1 \) will have converged and this implies that the feedback signals will have a difference of \( a_{k-1} - b_{k-1} = -4\alpha \) as shown in section 6.5. If the path with \(-2\) is the correct one, then the feedback signal for the \(0\) path will be in error, and will be too large by \( 4\alpha \). This is equivalent to the \(0\) being replaced with a value of \(-4\alpha\). Hence, the value of \( e_k \) will be \( e_k = -2 - (-4\alpha) \).

Thus, for the example error event in fig 6.16 the error sequence is \( e_k = \{-2,-2 + 4\alpha\} \). As the error event distance is \( \sqrt{\sum_k e_k^2} \), the error event described has a distance of \( \sqrt{8\sqrt{1 - 2\alpha + 2\alpha^2}} \). All error events of length 2 have this form.
The probability of an error event occurring in the presence of correlated noise was given in section 2.7.3 as

\[
P(\text{Error Event}) = Q\left( \frac{\sum_{j=0}^{J} \frac{e_j^2}{\sigma_j^2}}{\sqrt{\sum_{j=0}^{J} e_j^2 R_n(0) + \sum_{j=0}^{J} \sum_{j\neq i}^{J} e_i e_j R_n(i-j)}} \right) \quad (6.54)
\]

Hence, for error events of length 2 the argument of the \( Q() \) function is

\[
\frac{8(1 - 2\alpha + 2\alpha^2) / 2}{\sqrt{8(1 - 2\alpha + 2\alpha^2)R_n(0) + 8(1 - 2\alpha)R_n(1)}} \quad (6.55)
\]

In order to simplify this expression, the term \( 8(1 - 2\alpha)R_n(1) \) will be approximated by \( 8(1 - 2\alpha + 2\alpha^2)R_n(1) \) as \( R_n(1) \) will be less than \( R_n(0) \) and \( \alpha^2 \) will be small. This allows the expression to be written as

\[
Q\left( \frac{\sqrt{8(1 - 2\alpha + 2\alpha^2)}}{2\sqrt{R_n(0) + R_n(1)}} \right) \quad (6.56)
\]

Fig 6.17 shows a minimum error event of length 3. The correct path is \( \{-2, 0, 2\} \). At time index \( k - 1 \) the difference is \( e_{k-1} = -2 - 0 \) as before.

At index \( k \), both paths are zero, but the error path will have a value of \( 4\alpha \) added in error, and hence \( e_k = 0 - (-4\alpha) \).

At index \( k + 1 \), the correct path has a value 2 and the incorrect path has a value 0. However, the error in the feedback signal will be \( a_k - b_k = -(4\alpha)(1 - \alpha) - 4\alpha = -4\alpha^2 \) and hence \( e_{k+1} = 2 - (0 + 4\alpha^2) \).

This error event is hence characterized with

\[
e_k = \{-2, 4\alpha, 2 - 4\alpha^2\} \quad (6.57)
\]

If the \( \alpha^2 \) term is ignored then this can be written as \( e_k = \{-2, 4\alpha, 2\} \)

If the value of \( a_k - b_k \) is approximated by alternating values of 0 and \( -4\alpha \) for consecutive runs of channel zeros, as discussed in section 6.5, then the error events will have the general form

\[
e_k = \pm\{2, 4\alpha, 0, \ldots, 4\alpha, 0, \ldots, 4\alpha, 0, 2 - 4\alpha\} \quad l \text{ even} \quad (6.58)
\]
and

\[ e_k = \pm \{2, 4\alpha, 0, \ldots, 4\alpha, 0, \ldots, 4\alpha, -2\} \quad l \text{ odd} \]  \hspace{1cm} (6.59)

Comparing this to the PR1 minimum distance error events as in eqn 6.53, the runs of zeros are replaced by runs of \(4\alpha, 0\) and the end of the even length error events are reduced to \(2 - 4\alpha\). The runs of \(4\alpha, 0\) increase the distance between the sequences but introduce correlation terms in eqn 6.54. In order to simplify the calculations, the effect of the correlation terms and the increased distance will be ignored.

It can also be noted that using the approximation to implement the the detector as in section 6.5 causes some error in the calculations when runs of zeros occur. This error will be compensated for by the larger distance between sequences caused by the \(4\alpha, 0, \ldots\) terms in runs of zero samples. This verifies that the approximation used will not significantly alter the performance of the detector as verified in the simulated results shown previously in fig 6.10.

The error terms can hence be simplified to

\[ e_k = \begin{cases} 
\pm \{2, 0, \ldots, 0, 2 - 4\alpha\} & l \text{ even} \\
\pm \{2, 0, \ldots, 0, -2\} & l \text{ odd}
\end{cases} \]  \hspace{1cm} (6.60)

Using these simplified expressions and the same probabilities of error events being supported and error per error event, the probability of error for this detector may be calculated as

\[ P_e(\text{OPDDC}) = \sum_{i=2}^{\infty} \frac{2i}{2!} Q \left( \frac{\sqrt{2}}{\sqrt{R_0(0) - R_0(i)}} \right) + \sum_{i=1}^{\infty} \frac{2i}{2!} Q \left( \frac{\sqrt{2} \sqrt{1 - 2\alpha + 2\alpha^2}}{\sqrt{R_0(0) + R_0(i)}} \right) \]  \hspace{1cm} (6.61)

Fig 6.18 shows the calculated performance of the OPDDC detector and the simulated performance of the detector under electronic noise conditions. There is reasonable agreement between the calculated and simulated values considering that the effect of error propagation is neglected in the theoretical calculations.

Eqn 6.61 shows that in effect, the even length error events suffer a reduction in effective minimum distance by a factor of \(\sqrt{1 - 2\alpha + 2\alpha^2}\).

The choice of the parameter \(\alpha\) is hence a tradeoff between reducing the equalization noise enhancement by increasing \(\alpha\) and the associated loss in effective minimum distance of the detector.

6.8.1 Sensitivity to \(\alpha\) Parameter

Efficient implementation of the detector requires that the parameter \(\alpha\) in the feedback filter is chosen as a negative power of 2. The bit error rate of the OPDDC detector as a
function of $\alpha$ has been calculated based on the error probability expression in eqn 6.61. Fig 6.19 and fig 6.20 show the results. From these graphs, it can be seen that the value of $\alpha$ for maximum effective SNR is on a broad peak. Hence, choosing $\alpha$ as a negative power of 2 results in an insignificant SNR loss. These graphs confirm that at the densities of interest, the choice of $\alpha$ as a negative power of 2 does not severely limit the detector performance at high densities.

Figure 6.19: Sensitivity of detector to $\alpha$ parameter at densities of 2.5 and 2.75
6.8.2 Comparison with PR4 and EPR4 Detectors

Using the developed expressions for the probability of error of the OPDDC detector with the channel model described in chapter 2, the performance of this detector as a function of recording density can be calculated. Fig 6.21 plots the required channel SNR required to achieve a bit error rate of $10^{-6}$ on the Lorentz channel with electronics noise. The calculated performance of the detectors show that the OPDDC detector performs better than EPR4 for densities greater than 2.3. The OPDDC detector requires 1dB less SNR at a density of 3.0 than the EPR4 detector. This confirms that the target response and the detector perform well on high density recording channels.

Figure 6.20: Sensitivity of detector to $\alpha$ parameter at densities of 3.0 and 3.25

Figure 6.21: Required channel SNR for a BER of $10^{-6}$ for PR4, EPR4 and OPDDC
6.9 Simulated Performance of Detector

The performance of the OPDDC detector using the updated equations as in section 6.5.1 was simulated using the computer channel model described in chapter 2. The PR4, EPR4, DFE, FDTS/DF (using 2-sample tree search) and MLSD detection schemes were also simulated for comparison purposes.

Fig 6.22 show the results for white noise at the channel output with no medium noise. Table 6.10 shows the values of \( \alpha \) used at various densities. The channel SNR required for a bit error rate of \( 1 \times 10^{-6} \) is shown as a function of density. At densities above 2.3 bits per \( PW_{50} \), the OPDDC detector outperforms the EPR4 detector. At densities above 2.6, the OPDDC detector outperforms the FDTS/DF detector. At a density of 3.0 bits per \( PW_{50} \), the OPDDC detector performs 0.3 dB better than FDTS/DF and 0.8 dB better than EPR4. The OPDDC detector performs within 1.2 dB of the MLSD detector at a density of 2.8 bits per \( PW_{50} \).

![Graph showing required SNR for BER of 1e-6 in dB vs. recording density (bits per PW50)](image)

Figure 6.22: Comparison of simulated detector performance, white noise only

Fig 6.23 shows the results for white noise at the channel output and an equal amount of medium noise. The channel SNR required to achieved a bit error rate of \( 1 \times 10^{-6} \) is plotted. The performance of the OPDDC detector over existing detection schemes is similar to the white noise case. The OPDDC detector outperforms the EPR4 detector at densities above
<table>
<thead>
<tr>
<th>Density</th>
<th>$\alpha$ for $8/9 ,(0,4/4)$ Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>0.25</td>
</tr>
<tr>
<td>2.0</td>
<td>0.25</td>
</tr>
<tr>
<td>2.5</td>
<td>0.25</td>
</tr>
<tr>
<td>3.0</td>
<td>0.125</td>
</tr>
<tr>
<td>3.5</td>
<td>0.0625</td>
</tr>
</tbody>
</table>

Table 6.10: Value for $\alpha$ at different densities

2.2 bits per $PW_{50}$ and the FDTS/DF 2 sample detector at densities above 2.5 bits per $PW_{50}$. The OPDDC detector performs within 0.8 $dB$ of the MLSD detector at a density of 3.0 bits per $PW_{50}$ showing its suitability for high density recording.

![Graph](image)

Figure 6.23: Comparison of simulated detector performance with media noise.

6.10 Comparison with Other Detectors

The OPDDC detector and channel combine the properties of a number of detection schemes to implement a detector that performs well at high densities ($\approx 3$) and retains moderate complexity.

The equalization requirements for the OPDDC channel requires some amplitude equalization but not as severe as required for PR4 and EPR4 channels. It also requires consid-
erable phase equalization as in the case of a DFE or FDTS/DFE detector. Existing work on the DFE front end would be applicable to the OPDDC detector[MK95].

The timing recovery, gain control and adaption loops can be based on PR4 samples generated as described in section 6.6. In this way, these loops are based on preliminary decisions that do not require decisions from the main detector as in the case of a DFE or FDTS/DF system.

While the PR4 detector is readily implemented as two interleaved detectors operating at half the channel symbol rate, the EPR4 detector has 8 states and must operate at the full data rate requiring considerable area and power dissipation. The DFE implementation is more reasonable although it requires RAM lookups for the feedback calculation. The FDTS/DF is complex to implement as the detector for an uncoded channel requires a multiplier inside the feedback loop[CK91]. The OPDDC detector, while more complex than a PR4 or DFE detector, provides higher performance than these and is less complex to implement than an EPR4 detector or FDTS/DF detector. It also provides a range of complexity verses speed options that allows additional flexibly.

6.11 Conclusions

A novel channel target and corresponding detection scheme have been developed that combines the actions of decision feedback and maximum likelihood detection. The detector is novel in its use of an IIR feedback filter and Viterbi detector combined with decision feedback resulting in low computational requirements.

An associated method for operating timing recovery and control loops has been developed that does not depend on the main detector operation. New high rate line codes can be developed if required or existing line codes may be used.

Theoretical analysis and computer simulations confirm the performance of the detector on the magnetic recording channel.

The target response is particularly suited to high density recording channels and outperforms EPR4 detectors at densities above 2.2 bits per $PW_{50}$. The detector also outperforms FDTS/DF with 2 samples at densities above 2.5 bits per $PW_{50}$.

This new target and detector provide a new option for the design of future high density recording systems.
Chapter 7

Summary and Conclusions

7.1 Introduction

This thesis concerns the coding, channel equalization and detection processes for digital magnetic recording. Magnetic recording is a complex process, but, with the constraint that the recorded signal is a binary level signal, the system may be regarded as a linear communications channel suffering from severe ISI.

In this thesis existing and proposed schemes for future magnetic recording channels have been considered and a number of modifications and new detection methods have been proposed. The new schemes have been assessed based on their performance on a Lorentz channel model of the recording channel and on their practical implementation.

7.2 Context of Thesis

The PR4 partial response channel with Viterbi detection has been considered as a baseline system due to its dominance of the sampled data channel industry. This detector is most suitable for recording densities up to 2 bits/$PW_{50}$.

The EPR4 channel is recognized as a higher performance channel for higher recording densities, but with a large increase in complexity. A number of manufacturers are considering the EPR4 detector [SYM+93] with a commercial implementation recently released [Ana95]. The basic DFE is an alternative that is beginning to find some acceptance with a commercial product recently being released [Gra95].

Beyond these schemes, various more advanced detectors have been proposed. Trellis coding in the form of matched spectral null codes have been demonstrated to improve the performance on the PR4 channel[TRS+92][RCS+95]. The original MSN detectors suffered from long error events and path memories. Recently, time varying structures similar to the method proposed in chapter 3 and implemented in chapter 4 have been designed. Their advantage has been practically demonstrated. However, they do require an increased data
channel rate resulting in higher recording densities without an increase in user data rate due to the code rate loss.

More advanced forms of DFE architectures such as the FDTS/DF method have been proposed for future high density recording channels. However, the implementation complexity of such schemes pose a number of difficulties. Analog implementation of such detectors have recently been considered in order to allow feasible implementation [CBMP95].

7.3 Contributions of the Thesis

7.3.1 MSN Codes

The use of time varying trellises for the efficient implementation of matched spectral coding has been proposed and developed in chapter 3. A VLSI implementation has been designed and fabricated as described in chapter 4.

The use of MSN codes with a PR1 channel has been proposed in chapter 3 as a means of significantly increasing the user recording density on the magnetic recording channel. This should be of significant interest in helical scan magnetic recording where a DC free code is already required and hence, little or no rate loss may be incurred through the use of MSN coding techniques. The use of contact recording on such systems may allow the use of the higher recording densities required for the PR1 channel with MSN codes.

7.3.2 DL Codes

In chapter 5, a new type of coding scheme has been developed that provides a moderate coding gain on the PR4 channel but with the same code rates as existing channels. These distance lengthening (DL) codes provide an attractive alternative to MSN codes as the channel rate and recording density are unaltered. Using the techniques presented, a gain in effective SNR of up to $\approx 1.6dB$ may be obtained at current recording densities with no modification to the magnetic components or user data rate.

A detailed example of a rate 16/18 code that is readily implementable was developed. This code also uses a novel correction technique to limit the effect of any single minimum distance error event in a single 18 bit codeword. This eases the design of higher level ECC systems.

The Viterbi detector required is a time varying 6-state detector that can be readily deinterleaved as with the normal PR4 detectors. The complexity of the detector is similar to the complexity of the implemented MSN detector presented in chapter 4, and this shows that practical VLSI implementation of the method is achievable.

The method presented offers an incremental step between existing systems and the
higher complexity of advanced detection schemes which require major changes in channel architectures.

7.3.3 New High Density Target Response OPDDC

In chapter 6, a new target response combining the PR1 partial response and decision feedback techniques is proposed. The new target called one plus D with DC feedback (OPDDC) is particularly suitable for high density recording of \( \approx 3 \) bits/\( PW_{50} \). The required detector can be implemented using a 2-state Viterbi detector with decision feedback. Due to the nature of the target response and with some detailed observation of the detector operation, it has been possible to simplify the detector implementation with virtually no loss of performance. The practical implementation of the detector requires no multipliers and can be considerably pipelined for high speed VLSI implementation.

Other practical aspects of a channel using this target response have also been considered. In particular, a novel method has been proposed for the timing and control loops which allow loop operation before the main detector is operating.

7.4 Potential Applications of Proposed Schemes

The design and implementation of read channel architectures and their use depends on a large number of details. The error rate performance of the detection system is of primary importance. Practical VLSI implementation at the required data rates is also vital. These two issues have been a focus of this thesis. Other issues are more dependent on the application details.

In tape drives for archival storage, only sequential access is required. This allows the drives to operate at relatively high raw channel error rates and to use powerful coding techniques across multiple data blocks. They can use contact recording as the recording medium is written and readback infrequently relative to other devices. The use of contact recording also allows a high linear recording density. Such a regime suggests the applicability of a soft decision coding scheme like MSN codes with their associated rate loss. The use of the PR1 channel with MSN coding as proposed in chapter 3, provides a significant gain under these conditions.

In disk drives, data sectors must be readable using random access. The raw error rate from the channel must be quite low as all the ECC coding for a single sector must be contained within the sector. In this application, the transfer rate is also of primary importance. This requires having detection schemes that have efficient VLSI implementations that maximize the data rate while minimizing complexity and power dissipation. The trend in the disk drive industry is to use as high a code rate as possible as this maximizes the
user transfer rate for a given channel transfer rate. As the read head is not in contact with the medium, the linear density is more difficult to increase than in contact recording. This reduces the attractiveness of coding schemes with significant rate losses such as MSN codes. The new DL coding schemes proposed in chapter 5 address this issue by providing a coding gain with no rate loss over existing schemes. The high speed implementation of such detectors is readily achievable.

The new target and detector proposed in chapter 6 provides a detector with a superior high density error rate performance. High speed implementation of the detector and the ability to use high rate codes should be attractive for future disk drive products with higher recording densities.

7.5 Comparison of Existing and Proposed Detectors

Fig 7.1 shows the performance of the existing detectors and the detectors proposed in this thesis. The channel SNR required for a bit error rate of $1 \times 10^{-6}$ under equal electronics and medium noise is plotted against the recording density. Each detection system has an 8/9 code rate.

From the graph, it can be seen that below a recording density of 2.5 bits per $PW_{50}$, the RDS modulo 3 code proposed in chapter 5 performs better than the existing detectors.

At a density of 2.5 bits per $PW_{50}$ the RDS modulo 3 code, the OPDDC detector and the FDTS/DF 2 sample detector all perform at a similar level.

Above a recording density of 2.5 bits per $PW_{50}$, the OPDDC target and detector proposed in chapter 6 outperforms the existing detectors.

Table 7.1 tabulates the major attributes of the existing and proposed detection schemes and the new schemes developed in this thesis. From the table, it can be seen that the new methods proposed provide a number of options for existing and future channel designs and compare favorably in terms of complexity and performance.
<table>
<thead>
<tr>
<th>Performance</th>
<th>Complexity</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR4</td>
<td>Baseline for disk drive read channel</td>
<td>Industry standard</td>
</tr>
<tr>
<td>EPR4</td>
<td>Better than PR4 at densities above 2.0.</td>
<td>Commercial realization recently available</td>
</tr>
<tr>
<td>DFE</td>
<td>Slightly better performance than PR4.</td>
<td>Commercial realization recently available</td>
</tr>
<tr>
<td>FDTS/DF</td>
<td>Better than DFE at higher densities.</td>
<td>No commercial or known implementations available.</td>
</tr>
<tr>
<td>PR4 MSN</td>
<td>Better than PR4.</td>
<td>Experimental prototypes exist. Code rate loss over existing channels.</td>
</tr>
<tr>
<td>PR1 MSN</td>
<td>Good performance at high densities</td>
<td>Suitable for DC free helical scan recording.</td>
</tr>
<tr>
<td>DL codes</td>
<td>1.6dB better than PR4 system. Outperforms other detectors up to density of 2.5</td>
<td>Suitable for incremental development of existing PR4 channels. Fits well into existing PR4 channel architectures.</td>
</tr>
<tr>
<td>OPDDC</td>
<td>Outperforms other detectors at high density</td>
<td>Range of speed vs complexity options.</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison of Detection Methods
7.6 Future Work

In this thesis, the practical implementation of MSN codes has been developed and a VLSI implementation designed and fabricated. The use of MSN codes with the PR1 channel has been proposed. A novel coding scheme for use with the PR4 channel has been developed and a detailed example of an encoder/decoder presented. A novel new equalization target response and efficient detector have also been developed.

These schemes have been verified by analysis and simulation using a computer software model of the Lorentz channel model developed.

The most immediate objective of future work is to characterize the performance of the proposed schemes under the realistic conditions of a magnetic recording channel. This can be most easily achieved with access to a spinstand where sample patterns could be written on a suitable medium and the readback signal digitized. The digitized data could then be fed through the existing software models developed and the performance of the schemes analyzed in this way.

The following future work is envisaged:

1. The computer model developed has been used to model the error rate performance of existing and proposed detection systems. For the work presented in this thesis,
the computer model was optimized to measure the error rate performance of various
detection schemes with perfect timing recovery and equalization. However, the model
is flexible and has facilities to allow simulation with more realistic continuous time
filters and practical FIR filters. Timing recovery could be readily added and the com-
puter model used to evaluate trade offs between various implementation constraints.
Examples include the number of FIR taps required and their precision, the number
of poles and zeros required by the continuous time filter and the type of response and
programmability required.

2. The use of MSN codes with PR1 channels is most suitable for tape drives based on
helical scan technology where DC free codes are presently used. This includes the
DDS R-DAT tape drives and the DDS3 standard currently embodies the PR1 partial
response[FOH95]. When drive mechanism, heads and tape for this standard become
available, they provide an ideal platform for investigating the use of MSN codes as
described in chapter 3.

3. The DL codes presented in chapter 5 have the most immediate potential due to the
wide spread adoption of the PR4 target response in the disk drive industry. The
fact that they have the same rate as existing codes allows the possibility of modifying
an existing channel chip by changing the encoder/decoder logic, Viterbi detector
and synchronization circuits. The rest of the system would be unchanged and the
differences, from the designers perspective, would be small. The additional SNR
performance could be used to increase the linear recording density or track density,
or to allow more margin thus increasing production yields.

The weakest point of the developed codes is the possibility of long runs of zero channel
samples. The performance of existing timing recovery loops needs to be considered.
If timing jitter is a problem, then the loop bandwidth can be reduced or further
enhancements to the codes may be considered.

4. The OPDDC channel proposed in chapter 6 provides a technology for high density
recording in the future. The target is particularly suitable for recording densities
around 3 bits per $PW_{50}$. All aspects of a channel based on the proposed target have
been considered and it requires substantial changes from existing PRML channels.
However, the presence of a commercial DFE channel in the hard disk drive industry
indicates the possible acceptance of the technique. The front end equalizer for the
OPDDC would be quite similar to the DFE channel equalizer and both must contend
with error propagation. Existing work on the DFE front end would be applicable to
the OPDDC detector[MK95].

172
The actual detector implementation needs to be implemented in VLSI. The issues for VLSI implementation such as complexity and pipelining have been considered in chapter 6 and a VLSI design should be readily achieved. There is a range of complexity versus speed options possible and considerable scope for optimizing the design for a given process.

Characterizing the method on a magnetic recording channel is the most important outstanding issue. This would require access to the head media combinations suitable for very high recording densities. Sampled data waveforms from such systems at the proposed recording density would allow the use of the software model to assess in detail the potential gains from the proposed detector.

7.7 Conclusions

In this thesis, new and improved practical coding and detection methods for the magnetic recording channel have been developed. These include improvements and modifications to trellis coding through the use of matched spectral null codes, development of a novel coding method for the PR4 channel and a novel equalization target and detector for high density magnetic recording. The performance and complexity of the proposed schemes have been compared to existing detectors and shown to compare favorably. It is concluded that the proposed methods provide a new range of options for future magnetic recording channels. They allow incremental development of existing channels and technology for future high density recording.
References


Appendix A

Authors Publications


Appendix B

Mathematical Derivations

B.1 Frequency Response of the Lorentz Channel

Consider the function

\[ H(f) = e^{-\alpha|f|} \quad \alpha > 0 \]

The inverse Fourier transform of \( H(f) \) is

\[
\mathcal{F}^{-1}H(f) = \int_{-\infty}^{\infty} H(f)e^{i2\pi ft} df \\
= \int_{-\infty}^{0} e^{-\alpha|f|}e^{i2\pi ft} df \\
= \int_{-\infty}^{0} e^{\alpha f\cos t}e^{i2\pi ft} df + \int_{0}^{\infty} e^{-\alpha f\cos t}e^{i2\pi ft} df \\
= \int_{-\infty}^{0} e^{\alpha f\cos t}e^{i2\pi ft} df + \int_{0}^{\infty} e^{-\alpha f\cos t}e^{i2\pi ft} df \\
= \frac{1}{\alpha + j^2\pi t}e^{\alpha f\cos t}|_{0}^{-\infty} + \frac{1}{-\alpha + j^2\pi t}e^{-\alpha f\cos t}|_{0}^{\infty} \\
= \frac{1}{\alpha + j^2\pi t} - \frac{1}{-\alpha + j^2\pi t} \\
= \frac{2\alpha}{\alpha^2 + 4\pi^2 t^2} \tag{B.1}
\]

and hence

\[ \frac{2\alpha}{\alpha^2 + 4\pi^2 t^2} \xrightarrow{\mathcal{F}} e^{-\alpha|f|} \quad \alpha > 0 \tag{B.2} \]

The step response of the Lorentz channel model is

\[
h_{\text{lorz}}(t) = \frac{1}{1 + \frac{4\pi^2}{PW_{50}}} \\
= \frac{(2\pi)^2(PW_{50}^2/4)}{(2\pi)^2(PW_{50}^2/4) + (2\pi)^2t^2} \\
= \frac{\pi PW_{50}}{2\pi PW_{50}^2} \frac{2\pi PW_{50}}{(\pi PW_{50})^2 + (2\pi)^2} \tag{B.3}
\]

by letting \( \alpha = \pi PW_{50} \) in eqn B.2 the Fourier transform of the Lorentz function is

\[ h_{\text{lorz}}(t) \xrightarrow{\mathcal{F}} \frac{\pi PW_{50}}{2}e^{-\pi PW_{50}|f|} \tag{B.4} \]

B1
The impulse response of the Lorentz channel is the step response differentiated and hence the frequency response of the Lorentz channel is

\[
H_{\text{mag}}(f) = j2\pi f H(f) = j2\pi f \frac{PW_{50}}{2} e^{-\pi PW_{50}|f|}
\]  

(B.5)

B.2 Probability of Error for EPR4 Detector

The minimum distance error events for the EPR4 detector were derived in chapter 2 as follows.

For events with a length \( l \), the error events have the form shown in table B.1.

<table>
<thead>
<tr>
<th>Length</th>
<th>P(\text{Event Supported})</th>
<th>Bit Errors</th>
<th>( e_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l ) Odd</td>
<td>( \frac{16}{2^l} )</td>
<td>( l - 3 )</td>
<td>( \pm {+2, 0, -2, 0, \ldots, 0, -2, 0, +2} )</td>
</tr>
<tr>
<td>( l ) Even</td>
<td>( \frac{2^{l+1/2}}{2^l} )</td>
<td>( \frac{l}{2} - 1 )</td>
<td>( \pm {+2, +2, 0, \ldots, 0, -2, -2} )</td>
</tr>
<tr>
<td></td>
<td>( \frac{16}{2^l} )</td>
<td>( l - 3 )</td>
<td>( \pm {+2, 0, -2, 0, \ldots, 0, +2, 0, -2} )</td>
</tr>
</tbody>
</table>

Table B.1: Error Events for EPR4 detector

For events of the form \( \{+2, 0, -2, 0, \ldots, 0, -2, 0, +2\} \),

\[
\sum_k e_k^2 = 16 \tag{B.6}
\]

and

\[
\sum_k e_k^2 R_n(0) + \sum_i \sum_{j \neq i} e_i e_j R_n(i - j) = 16 R_n(0) + 2e_0 e_2 R_n(2) + 2e_0 e_{l-3} R_n(l - 3) + 2e_0 e_{l-1} R_n(l - 1) + 2e_{l-3} e_{l-1} R_n(2)
\]

\[= 16 R_n(0) - 16 R_n(2) - 16 R_n(l - 3) + 8 R_n(l - 5) + 8 R_n(l - 1) \tag{B.7}
\]
Hence the probability of this error event occuring is (eqn 2.64)

\[
P(e_k) = Q \left( \frac{\sum_{k=0}^{J} \frac{e_k^2}{R_n}}{\sqrt{\sum_{k=0}^{J} e_k^2 R_n(0) + \sum_{i,j} \sum_{j=0,j\neq i}^{J} e_i e_j R_n(i-j)}} \right)
\]

\[
= Q \left( \frac{16/2}{\sqrt{16R_n(0) - 16R_n(2) - 16R_n(l-3) + 8R_n(l-5) + 8R_n(l-1)}} \right)
\]

\[
= Q \left( \frac{2}{\sqrt{R_n(0) - R_n(2) - R_n(l-3) + R_n(l-5)/2 + R_n(l-1)/2}} \right)
\]

(B.8)

For events of the form \(\pm \{+2, 0, -2, 0, \ldots, 0, +2, 0, -2\}\),
\[
\sum_{k} e_k^2 = 16
\]

(B.9)

and

\[
\sum_{k} e_k^2 R_n(0) + \sum_{i} \sum_{j=0,j\neq i}^{J} e_i e_j R_n(i-j) = 16R_n(0) + 16R_n(2) + 16R_n(l-3) - 8R_n(l-5) - 8R_n(l-1)
\]

(B.10)

Hence the probability of this error event occuring is (eqn 2.64)

\[
P(e_k) = Q \left( \frac{\sum_{k=0}^{J} \frac{e_k^2}{R_n}}{\sqrt{\sum_{k=0}^{J} e_k^2 R_n(0) + \sum_{i,j} \sum_{j=0,j\neq i}^{J} e_i e_j R_n(i-j)}} \right)
\]

\[
= Q \left( \frac{16/2}{\sqrt{16R_n(0) - 16R_n(2) + 16R_n(l-3) - 8R_n(l-5) - 8R_n(l-1)}} \right)
\]

\[
= Q \left( \frac{2}{\sqrt{R_n(0) - R_n(2) + R_n(l-3) - R_n(l-5)/2 - R_n(l-1)/2}} \right)
\]

(B.11)

Similarly for events of the form \(\pm \{+2, +2, 0, \ldots, 0, -2, -2\}\),
\[
\sum_{k} e_k^2 = 16
\]

(B.12)
and

\[
\sum_k e_k^2 R_n(0) + \sum_i \sum_{j \neq i} e_i e_j R_n(i - j) = 16R_n(0) \\
+ 2e_0 e_1 R_n(1) + 2e_0 e_{l-2} R_n(l - 2) + 2e_0 e_{l-1} R_n(l - 1) \\
+ 2e_1 e_{l-2} R_n(l - 3) + 2e_1 e_{l-1} R_n(l - 2) \\
+ 2e_{l-2} e_{l-1} R_n(1) \\
= 16R_n(0) + 16R_n(1) - 16R_n(l - 2) - 8R_n(l - 3) - 8R_n(l - 1)
\]  

(B.13)

Hence the probability of this error event occurring is

\[
P(e_k) = Q \left( \frac{\sum_{k=0}^{l-1} e_k^2}{\sqrt{\sum_{k=0}^{l-1} e_k^2 R_n(0) + \sum_{k=0}^{l-1} \sum_{j=0, j \neq i} e_i e_j R_n(i - j)}} \right) \\
= Q \left( \frac{16/2}{\sqrt{16R_n(0) + 16R_n(1) - 16R_n(l - 2) - 8R_n(l - 3) - 8R_n(l - 1)}} \right) \\
= Q \left( \frac{2}{\sqrt{R_n(0) + R_n(1) - R_n(l - 2) - R_n(l - 3)/2 - R_n(l - 1)/2}} \right)
\]  

(B.14)

Using eqn B.8, B.11 and eqn B.14 for the probability of error events and the probability of error events being supported and the number of bit errors per error event from table B.1 in eqn 2.65, the probability of error for the EPR4 detector becomes

\[
P_e(EPR4) = Q \left( \frac{2}{\sqrt{R_n(0) + R_n(1)/2 - R_n(2) - R_n(3)/2}} \right) \\
+ \sum_{l=7}^{l_{even}} \frac{16}{2^l} (l - 3) Q \left( \frac{2}{\sqrt{R_n(0) - R_n(2) - R_n((l - 3)) + R_n(l - 5)/2 + R_n(l - 1)/2}} \right) \\
+ \sum_{l=0}^{\infty} \frac{4}{2^l/2 (l/2 - 1)} Q \left( \frac{2}{\sqrt{R_n(0) + R_n(1) - R_n(l - 2) - R_n(l - 3)/2 - R_n(l - 1)/2}} \right) \\
+ \frac{16}{2^l} (l - 3) Q \left( \frac{2}{\sqrt{R_n(0) - R_n(2) + R_n(l - 3) - R_n(l - 5)/2 - R_n(l - 1)/2}} \right)
\]  

(B.15)
Appendix C

Channel Model Simulation Software

C.1 Software Description

The channel simulation software *CMOD* is based on a number of C programmed modules that perform the various functions required. The software has been designed for the purpose of error rate measurements.

The operation of the program starts with an initialization section that setups the parameters required for a simulation.

The code then passes 8000 bits through the channel during which time the channel delay is measured and the LMS filter is adapted. The LMS filter taps are then written to a file.

The main program loop is then iterated passing data through the channel and detecting it. The main loop may pass a fixed number of bits through the channel while writing signals at various points to a file for post processing. Alternatively, the loop may iterate until a predefined number of errors occur. The error rate may then be computed based on the number of bits transmitted. The actual errors and their location may be written to a file for later analysis of error event lengths and error propagation.

These functions may then be repeated while sweeping any parameters required and changing the detection system.

The data files generated are normal UNIX text files with simple data formats. Standard UNIX tools may be used to post process the data. A number of command line programs have also been written to help in post processing the data. Alternatively, the files could be read into a number of commercial mathematic packages for further analysis.

The software was written to be used in batch mode where the simulations to be performed are predefined and the software is left to run without user interaction. Fig C.1 shows a flow diagram of the main loops in the software.
C.1.1 Platforms

The code has been developed under the LINUX operating system, kernel 1.2.1, on a PC compatible machine. The source is written in C and requires an ANSI compatible
compiler. Under LINUX the GCC\textsuperscript{1} compiler has been used. The source should be portable to most other UNIX systems that have an ANSI compatible compiler or preferable GCC. It has been successfully compiled on a Decstation 5200 under Ultrix V4.1 and on a Sun Sparc Station 5 with GCC. Porting to other non-Unix platforms should be straightforward although 32 bit integers are assumed in the code.

C.1.2 Performance

The performance of the software on a PC compatible machine with a 133Mhz Pentium Processor was measured. The software calculated \( \approx 2500 \) channel bits per second.

C.2 Software Functionality

C.2.1 Lorentz Channel and Basic Parameters

The software implements a model of the linear Lorentz channel model with both electronic and media noise. The parameters for the channel model are shown in table C.1.

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Parameter} & \textbf{Description} \\
\hline
\hline
\textit{PW}_{50} & Lorentz channel 50\% pulse width. \\
\hline
\textit{T}_{bit} & Recorded bit period. \\
\hline
\textit{N}_{elect} & Electronics noise power spectral density \\
\hline
\textit{N}_{media} & Media noise power spectral density \\
\hline
\textit{LMS} & Filter length \\
\hline
\textit{Number of errors} & to wait for BER measurement \\
\hline
\end{tabular}
\end{center}

Table C.1: Channel Parameters

C.2.2 Data Source

The input data for the channel may come from a number of possible random data sources. The implemented sources are listed in table C.2.

C.2.3 Continuous time filter

The oversampled output of the channel model is filtered with a bilinear transform mapping of a continuous time filter. The implemented filters are listed in table C.3

\textsuperscript{1}GNU C Compiler
Table C.2: Data Sources

<table>
<thead>
<tr>
<th>Filter</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 Pole Bessel Low Pass</td>
<td>Cutoff Freq and Boost</td>
</tr>
<tr>
<td>6 Pole, 2 Zero Bessel Low Pass</td>
<td>Cutoff Freq, Zero positions</td>
</tr>
<tr>
<td>7 Pole, 2 Zero Equiripple Low Pass</td>
<td>Cutoff Freq, Zero positions</td>
</tr>
<tr>
<td>8 Pole Butterworth Low Pass filter</td>
<td>Cutoff frequency</td>
</tr>
<tr>
<td>90 degree Phase shifter approximation</td>
<td>Center frequency</td>
</tr>
</tbody>
</table>

Table C.3: Continuous Time Filters

C.2.4 Target Responses

The LMS filter adapts based on the correct data being fed through a target response.

The implemented target responses are listed in table C.4.

<table>
<thead>
<tr>
<th>Target Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR4</td>
</tr>
<tr>
<td>EPR4</td>
</tr>
<tr>
<td>PR1</td>
</tr>
<tr>
<td>FLat response</td>
</tr>
<tr>
<td>OPDCC</td>
</tr>
<tr>
<td>DFE</td>
</tr>
</tbody>
</table>

Table C.4: Target Responses

C.2.5 Detectors

The LMS filter output is fed to a detector to recover the transmitted data. The implemented detectors are listed in table C.5.
<table>
<thead>
<tr>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR4 Viterbi 32 bit path memory</td>
</tr>
<tr>
<td>EPR4 Viterbi 32 bit path memory</td>
</tr>
<tr>
<td>PR1 Viterbi 32 bit path memory</td>
</tr>
<tr>
<td>Rate 6/8 time varying detector, chapter 4</td>
</tr>
<tr>
<td>Rate 8/9 Single bit parity time varying detector</td>
</tr>
<tr>
<td>Rate 14/16 Double bit parity time varying detector</td>
</tr>
<tr>
<td>Rate 16/18 RDS modulo 3 code time varying detector</td>
</tr>
<tr>
<td>Binary Threshold detector</td>
</tr>
<tr>
<td>Rate 12/16 time varying Nyquist Null MSN</td>
</tr>
<tr>
<td>Rate 9/12 time varying Nyquist Null MSN</td>
</tr>
<tr>
<td>DFE</td>
</tr>
<tr>
<td>FDTS/DF 2 sample detector</td>
</tr>
<tr>
<td>OPDDC detector</td>
</tr>
</tbody>
</table>

Table C.5: Detectors
Appendix D

Bit Error Rate Confidence Interval

A typical method of measuring the bit error rate of a system would be to count the number of transmitted bits \( N \) and the number of errors \( E \) that occurred. The error rate would then be calculated as

\[
P_{\text{error}} \approx \frac{E}{N}
\]

Of interest is the confidence interval in the calculated error rate or the number of errors required to be observed to obtain a reasonable accuracy in the measurement.

Let the actual probability of error for the system be \( p \). If \( N \) bits are counted then the probability of \( E \) errors occurring is

\[
P(E, N, p) = p^E(1 - p)^{N - E} \binom{N}{E}
\]

where \( \binom{N}{E} \) is the number of ways of selecting \( E \) objects from \( N \) objects.

If \( E \) errors are counted in \( N \) bits then it is required to calculate the range of probabilities \( p \) that are likely to cause the observed number of errors. The total probability of the observed \( E \) errors is

\[
\int_0^1 P(E, N, p)dp
\]

It is required to find two probabilities \( p_1 \) and \( p_2 \) such that it is most likely that \( p \) is in this range. Hence, \( p_1 \) and \( p_2 \) are required to be determined with

\[
\frac{\int_{p_1}^{p_2} P(E, N, p)dp}{\int_0^1 P(E, N, p)dp} = C \tag{D.1}
\]

where \( C \) is the level of accuracy required, e.g. \( C = 0.95 \) for a 95% confidence level. The additional requirement that

\[
P(E, N, p_1) = P(E, N, p_2) \tag{D.2}
\]

is also chosen such that in a skewed distribution the probability of the limits \( p_1 \) and \( p_2 \) are equal.
Solving eqn D.1 and eqn D.2 numerically with a 95% confidence interval for $p_1$ and $p_2$, results in the plot of fig D.1. The effective SNR is defined as $20\log_{10} Q^{-1}(p)$.

Figure D.1: 95% Confidence Intervals with $E$ errors at different S/N ratios
Appendix E

MSN IC

E.1 Representative Schematics

This appendix shows some representative schematics from the rate 6/8 matched spectral null code chip described in chapter 4.

The examples shown are

Viterbi top level schematic This is the top level of the Viterbi detector section of the IC. It includes the 6 ACS units, Path memory and receive section.

ACS unit for state DK

ACS unit for state CH

Metric calculation for $\lambda_0$ This circuit adds the metric for $\lambda_0$ to the input 9 bit path metric and is used in all of the ACS units.

Metric calculation for $\lambda_{\pm2}$ This circuit adds the metric for $\lambda_{\pm2}$ or $\lambda_{-2}$ depending on an input control unit. This function is required in the EI BG AI and FJ ACS units.

Path Memory The complete path memory required for the Viterbi detector is shown.

Synthesized 6 to 8 encoder

Synthesized 8 to 6 decoder The synthesized schematics for the encoder and decoder are shown to illustrate the reasonably compact implementation of these blocks.
Figure E.1: Viterbi Detector top level schematic
Figure E.2: ACS unit for state **DK**
Figure E.6: Viterbi Path Memory
Figure E.8: Synthesized 8 to 6 decoder
Appendix F

Rate 16/18 RDS mod 3 code

F.1 VERILOG Description of Encoder

/ *
FILE: encs.v
DATE: 3/4/96
AUTHOR: Tom Conway (c) 1996
DESC:
  Encodes 16 user data bits to 18 codeword bits.
  Each codeword has a RDS mod 3 of 0
  Bit 0 should be transmitted first and bit 17 last.
  There is guaranteed to be a transition in bits [7:5]
  and [17:14].
*/

module enc1618( b, c );
input [15:0] b;
output [17:0] c;

wire [1:0] rds10;

assign c[4:0] = b[4:0];
assign c[13:9] = b[12:8];
assign rds10 = rdvalue10( {b[12:8],b[4:0]} );
assign {c[17:14],c[8:5]} = map6to8( b[15:13], b[7:5], rds10 );

function [7:0] map6to8;
input [2:0] bup, blo;
input [1:0] rdsin ;

reg [15:0] b;
reg [1:0] rdsval;

begin
  b[15:13] = bup;
  b[7:5] = blo;
endfunction

F 1
if ( {b[13],b[7:5]} == 4'b0000 ) begin
  map6to8[7:4] = 4'b1001;
  rdsvX = rsdin; /* plus zero for 1001 seq */
  case( rdsvX )
    0: map6to8[3:0] = tableA( b[15:14] );
    1: map6to8[3:0] = tableC( b[15:14] );
    2: map6to8[3:0] = tableB( b[15:14] );
    default: map6to8[3:0] = 4'bxxxx;
  endcase
end
else if ( {b[13],b[7:5]} == 4'b1111 ) begin
  map6to8[7:4] = 4'b0110;
  rdsvX = rsdin; /* plus zero for 0110 seq */
  case( rdsvX )
    0: map6to8[3:0] = tableA( b[15:14] );
    1: map6to8[3:0] = tableC( b[15:14] );
    2: map6to8[3:0] = tableB( b[15:14] );
    default: map6to8[3:0] = 4'bxxxx;
  endcase
end
else begin
  map6to8[3:0] = {b[13],b[7:5]};
  rdsvX = rdssum( rdsvA4( {b[13],b[7:5]} ), rsdin );
  case( rdsvX )
    0: map6to8[7:4] = tableA( b[15:14] );
    1: map6to8[7:4] = tableC( b[15:14] );
    2: map6to8[7:4] = tableB( b[15:14] );
    default: map6to8[7:4] = 4'bxxxx;
  endcase
end
end
endfunction

function [3:0] tableA; /* table A with RDS mod 3 seq of 0 */
  input [1:0] index;
  begin
    case (index)
      0: tableA = 4'b1100;
      1: tableA = 4'b1010;
      2: tableA = 4'b0101;
      3: tableA = 4'b0011;
    endcase
  end
endfunction

function [3:0] tableB; /* table C with RDS mod 3 seq of 1 */
  input [1:0] index;
  begin
    case (index)
      0: tableB = 4'b1000;
      1: tableB = 4'b0100;
  end
endfunction
2: tableB = 4'b0010;
3: tableB = 4'b0001;
endcase
end
endfunction

function [3:0] tableC; /* table B with RDS mod 3 seq of 2 */
input [1:0] index;
begin
  case (index)
    0: tableC = 4'b1110;
    1: tableC = 4'b1101;
    2: tableC = 4'b1011;
    3: tableC = 4'b0111;
  endcase
end
endfunction

function [1:0] rdsvalue4; /* Calculate RDS mod 3 of 4 bits */
input [3:0] din;
integer i;

begin
  rdsvalue4 = 0;
  for( i=0; i<4; i=i+1 ) begin
    rdsvalue4 = rdsadd(rdsvalue4, din[i]);
  end
end
endfunction /* End of rdsvalue4 */

function [1:0] rdsvalue10; /* Calculate RDS mod 3 of 10 bits */
input [9:0] din;
integer i;

begin
  rdsvalue10 = 0;
  for( i=0; i<10; i=i+1 ) begin
    rdsvalue10 = rdsadd( rdsvalue10, din[i] );
  end
end
endfunction /* End of rdsvalue10 */

function [1:0] rdsadd;
input [1:0] rdsin;
input bit;
begin
  if ( bit )
    case ( rdsin )

0: rdsadd = 1;
1: rdsadd = 2;
2: rdsadd = 0;
3: rdsadd = 2'bxx;
endcase
else
case ( rdsin )
0: rdsadd = 2;
1: rdsadd = 0;
2: rdsadd = 1;
3: rdsadd = 2'bxx;
endcase
end
endfunction

function [1:0] rdssum; /* Add +1 or -1 to rds value mod 3*/
inout [1:0] sa, sb;
begin
  case(sa)
    0: begin case (sb)
        0: rdssum = 0;
        1: rdssum = 1;
        2: rdssum = 2;
        3: rdssum = 2'bxx;
      endcase
    end
    1: begin case (sb)
        0: rdssum = 1;
        1: rdssum = 2;
        2: rdssum = 0;
        3: rdssum = 2'bxx;
      endcase
    end
    2: begin case (sb)
        0: rdssum = 2;
        1: rdssum = 0;
        2: rdssum = 1;
        3: rdssum = 2'bxx;
      endcase
    end
    3: rdssum = 2'bxx;
  endcase
end
endfunction
endmodule
F.2 VERILOG Description of Corrector

/*
 * FILE: cors.v
 * DATE:  3/4/96
 * AUTHOR: Tom Conway     (c) 1996
 * DESC:
 * Corrects 18 bit RDS mod 3 codeword.
 * Correction corrects minimum distance events that
 * start in the last transition group of a code word
 * i.e bits 17:14
 */

module correct( cin, cout );
input [17:0] cin;
output [17:0] cout;

wire [1:0] rds14;
assign rds14 = rdsvalue14( cin[13:0] );  /* RDS mod 3 value of first 14 bits */
assign cout[13:0] = cin[13:0];  /* Unaffected by correction */
assign cout[17:14] = correct( cin[17:14], rds14 );  /* bits to be corrected */

function [3:0] correct;
    input [3:0] c;
    input [1:0] rdsin;
    begin
        if ( rdsin == 0 ) begin  /* RDS of 4 bits must be 0 */
            correct = correcta( c );
        end
        else
            if ( rdsin == 1 ) begin  /* RDS of 4 bits must be 2 */
                correct = correctc( c );
            end
            else
                if ( rdsin == 2 ) begin  /* RDS of 4 bits must be 1 */
                    correct = correctb( c );
                end
                else correct = 4'bxxxx;
        end
    endfunction

function [3:0] correcta;  /* make correction for bit of set A */
    input [3:0] d;
    begin
        casex ( d )
        4'b11xx: correcta = 4'b1100;
        4'b101x: correcta = 4'b1010;
        4'b011x: correcta = 4'b0110;
        4'b100x: correcta = 4'b1001;
        4'b010x: correcta = 4'b0101;
    endcase
*/
        4'b00xx: correcta = 4'b0011;
        endcase
        end
endfunction

function [3:0] correctb; /* make correction for bit of set B */
     input [3:0] d;
     begin
        casex ( d )
        4'b1xxx: correctb = 4'b1000;
        4'b01xx: correctb = 4'b0100;
        4'b001x: correctb = 4'b0010;
        4'b000x: correctb = 4'b0001;
        endcase
        end
endfunction

function [3:0] correctc; /* make correction for bit of set C */
     input [3:0] d;
     begin
        casex ( d )
        4'b111x: correctc = 4'b1110;
        4'b110x: correctc = 4'b1101;
        4'b10xx: correctc = 4'b1011;
        4'b0xxx: correctc = 4'b0111;
        endcase
        end
endfunction

function [1:0] rdsvalue14;       /* Calculate RDS mod 3 of 14 bits */
     input [13:0] din;
     integer i;

     begin
         rdsvalue14 = 0;
         for( i=0; i<14; i=i+1 ) begin
             rdsvalue14 = rdsadd( rdsvalue14, din[i] );
         end
     end
endfunction                    /* End of rdsvalue14 */

function [1:0] rdsadd;        /* Add +1 or -1 to rds value mod 3*/
     input [1:0] rdsin;
     input bit;
     begin
         if ( bit )
             case ( rdsin )
                0: rdsadd = 1;
                1: rdsadd = 2;
                2: rdsadd = 0;
            endcase
        end
endfunction
3: rdsadd = 2'bxx;
endcase
else
  case ( rdsin )
  0: rdsadd = 2;
  1: rdsadd = 0;
  2: rdsadd = 1;
  3: rdsadd = 2'bxx;
endcase
end
endfunction /* End of rdsadd */

eendmodule

F.3  VERILOG Description of Decoder

/*
FILE:  decs.v
DATE:  3/4/96
AUTHOR:  Tom Conway   (c) 1996
DESC:
  Decodes 18 bit codeword to 16 bit user data.
  Bit 0 should be received first and bit 17 last.
*/

module dec1618( c, b );
output [15:0] b;
input [17:0] c;

assign b[4:0] = c[4:0];
assign b[12:8] = c[13:9];
assign {b[15:13],b[7:5]} = map8to6( c[17:14], c[8:5] );

function [5:0] map8to6;
input [3:0] cup, clo;

  reg [17:0] c;

  begin
    c[17:14] = cup;
    c[8:5]   = clo;

    if ( c[17:14] == 4'b1001 ) begin
      map8to6[3:0] = 4'b0000;
      map8to6[5:4] = tabdec( c[8:5] );
    end

    else
      map8to6[4:0] = c[4:0];

    end

endfunction

endmodule
else if ( c[17:14] == 4'd0110 ) begin
    map8to6[3:0] = 4'd1111;
    map8to6[5:4] = tabdec( c[8:5] );
end
else begin
    map8to6[3:0] = c[8:5];
    map8to6[5:4] = tabdec( c[17:14] );
end
endfunction

function [1:0] tabdec;  /* Decode index from tables A,B and C */
    input [3:0] d;
begin
    case (d)
        4'd1100: tabdec = 0;
        4'd1010: tabdec = 1;
        4'd0101: tabdec = 2;
        4'd0011: tabdec = 3;
        4'd1000: tabdec = 0;
        4'd0100: tabdec = 1;
        4'd0010: tabdec = 2;
        4'd0001: tabdec = 3;
        4'd1110: tabdec = 0;
        4'd1101: tabdec = 1;
        4'd1011: tabdec = 2;
        4'd0111: tabdec = 3;
        default: tabdec = 2'dxx;
    endcase
end
endfunction
endmodule
F.4 Synthesized Logic

The Verilog descriptions were synthesized to generate logic level circuits for the rate 16/18 modulo 3 encoder, corrector and decoder. The resulting schematics are shown in figs F.1, F.2 and F.3, and show that practical implementation is readily achievable.

Figure F.1: Rate 16/18 encoder Logic
Figure F.2: Rate 16/18 code correction Logic

Figure F.3: Rate 16/18 Decoder