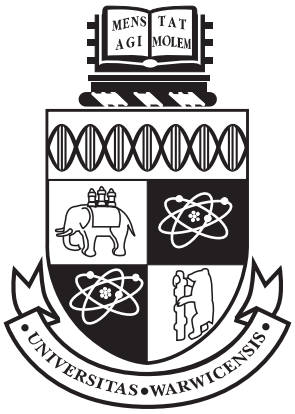
Data Detection Algorithms for Perpendicular Magnetic Recording in the Presence of Strong Media Noise



by

Robert Charles Jackson, BSc

A thesis submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Mathematics

University of Warwick

December 2008

Contents

1 Viterbi Fundamentals **19**

1.1 Introduction 19

1.2 Maximum Likelihood Detection 21

1.3 White Noise Viterbi Detector 35

2 High Throughput Viterbi Detectors **38**

2.1 Trellis Unrolling 39

2.2 ACS Retiming 44

2.3 ACS Bit Level Pipelining 46

2.4 Loop Elimination 47

2.4.1 Loop Elimination Complexity 56

2.5 Asymptotic Complexity of PMU 57

2.5.1 Complexity of Unrolled 1T Implementation 58

2.5.2 Complexity of K-T Implementation 59

2.6 Invariants 63

2.7 4-State White Noise Viterbi Detector Implementations 74

2.7.1 Standard 2T Implementation 75

2.7.2 3T Implementation with Loop Elimination 77

2.7.3 3T Implementation with Loop Elimination and Path Invariants 79

3 High Throughput Viterbi Decoders 84

3.1 Loop Invariants in Communications 85

3.2 Communications Example 100

4 Magnetic Channel 102

4.1 Read Channel Noise Model 102

4.2 Determining model parameters 104

4.2.1 Clean signal energy and representation as linear ISI . . 106

4.2.2 Media Noise 111

4.2.3 Isolated Transition - Error Function 115

4.2.4 Isolated Transition - Hyperbolic Tangent 117

5 Data Dependent Detectors 119

5.1 Auto-Regressive Noise Viterbi Detector 120

5.1.1 Computation of Noise Parameters 122

5.1.2 Simulation Results 124

5.1.3 Limitations 125

5.1.4 Equivalence To White Noise Detector 126

5.2 Data Dependent Auto-Regressive Noise Viterbi Detector . . . 127

5.2.1 Computation of Noise Parameters 129

5.2.2 Simulation Results 135

5.2.3 Limitations 136

5.3 Block Diagonal Detectors 137

5.3.1 Simulation Results 141
5.3.2 Limitations 142
5.4 Data Dependent Noise Predictive Detector 142
5.4.1 Simulation Results 143
5.4.2 Limitations 144
5.4.3 Reduced State ISI Predictive Detectors 145
5.5 Double Detectors 146
5.5.1 Simulation Results 148

6 Cost Function **150**
6.1 Choosing ISI target and equaliser coefficients 151
6.1.1 Maximum SNR Criteria 152
6.1.2 Minimum Mean Squared Error Criteria 155
6.1.3 Noise Spectra 158
6.1.4 MMSE Equivalence to Whitening Noise 166
6.1.5 Minimise Bit Error Rate Criteria 167

7 Binary Addition **185**
7.1 Introduction 185
7.2 Background 186
7.3 Ripple Carry Adder 188
7.4 Carry Select Adder 189
7.5 Parallel Prefix Adder 190
7.6 High Radix Parallel Prefix Adder 196
7.7 Ling Adder 197
7.8 Generalisation of Ling Adder 200

7.8.1 Generalised Ling Fundamentals 200
7.8.2 Modified Pseudo Generate and Hyper Propagate Functions 208
7.8.3 Generalised Ling Radix-3 Recursion 210
7.8.4 Generalised Ling Radix-3 Example 214
7.8.5 Higher Radix Generalised Ling Recursions 216
7.8.6 Generalised Ling Complexity 217
7.9 Summary 218

8 Conclusions **220**

8.1 Loop Elimination 220
8.2 Invariants 222
8.3 Data Dependent Double Detectors 223
8.4 Cost Function 225
8.5 Binary Addition 228

List of Figures

1.1 Low density waveform without ISI and high density waveform with ISI, for perpendicular magnetic recoding channel. 20

1.2 4 state trellis. 29

1.3 4 state trellis showing maximum likelihood path. 29

1.4 Two branches joining at a single state. 30

1.5 8 state (1,4) RLL trellis. 32

1.6 4 state trellis showing maximum likelihood path and convergence of contending paths. 34

2.1 Viterbi detector implementation comprising branch metric, path metric and traceback units. 39

2.2 Path metric unit comprising ACS units for each state. 39

2.3 1T ACS unit. 40

2.4 4 state 2T trellis. 41

2.5 2T ACS unit. 42

2.6 4 state 3T trellis. 42

2.7 Two iterations of 1T ACS unit, with 1T CSA retiming region indicated. 44

2.8 Unoptimised 1T CSA unit. 45
2.9 Optimised 1T CSA unit. 45
2.10 All paths converging at state 1 of 3 time slices of a 4 state trellis. 47
2.11 Exactly two paths connecting initial and final state. 47
2.12 3T ACS unit. 49
2.13 3T ACS unit with duplicated input path metrics. Red boxes indicate region to be transformed. 52
2.14 3T ACS unit with duplicated input path metrics, transformed to a 2T ACS unit (shown in black). 53
2.15 4 state trellis illustrating loop elimination. (a) shows original 1T trellis. (b) equivalent 3T trellis. (c) compare sides of each loop. (d) remove longest side of each loop. (e) 3T trellis with loops eliminated. 54
2.16 2 state trellis illustrating recursive loop elimination. 55
2.17 Two loops with same initial state and neighbouring final states. 67
2.18 Two loops with same final state and neighbouring initial states. 70
2.19 Two loops with same final state and any two initial states. . . 72
2.20 4 state 2T trellis. 75
2.21 4 state 3T trellis. 77

3.1 $K = 7$ rate $\frac{1}{2}$ convolutional code octal (171,133). 85

4.1 Isolated transition, isolated transition with position jitter and isolated transition with phase jitter, $T_{50} = 1.4$. 103
4.2 Measuring T_{50}. 116

5.1 Comparison of white noise MMSE detector and data independent auto-regressive detector. 125
5.2 Comparison of white noise MMSE detector, data independent auto-regressive detector and data dependent auto-regressive detector. 135
5.3 Comparison of data dependent auto-regressive detectors with different parameters. 137
5.4 Comparison of white noise MMSE detector and block diagonal detector and data dependent auto-regressive detector. 141
5.5 Local traceback of single data bit x_{N-K-1}. 143
5.6 Comparison of white noise MMSE detector, data dependent auto-regressive detector and data dependent noise predictive detector. 144
5.7 Comparison of 4 and 8 state white noise MMSE detectors against a 4 state ISI predictive white noise detector. 145
5.8 DDNP detector showing long noise predictive loop between ACS and BMU. 146
5.9 Double detector replaces noise predictive loop with survivor information from pre-detector. 147
5.10 Comparison of white noise MMSE detector, data dependent auto-regressive detector, data dependent noise predictive detector and double detector. 149

6.1 Spectrum of unequalised received signal for [1 3 3 1] AWGN channel. 158

6.2 Spectrum of received signal equalised to target [1 2.1 2.1 1] as chosen by SNR criteria for [1 3 3 1] AWGN channel. 159
6.3 Spectrum of received signal equalised to target [1 1.1 0.6 0.1] as chosen by MMSE criteria for [1 3 3 1] AWGN channel. . . . 159
6.4 Spectrum of unequalised received signal for 70% position jitter erf channel. 160
6.5 Spectrum of received signal equalised to target [1.00 2.21 2.21 1.00] as chosen by SNR criteria for 70% position jitter erf channel. 161
6.6 Spectrum of received signal equalised to target [1.00 0.97 0.32 -0.04] as chosen by MMSE criteria for 70% position jitter erf channel. 161
6.7 Spectrum of unequalised received signal for 90% media noise 90% position jitter erf channel. 162
6.8 Spectrum of received signal equalised to target [1.00 2.23 2.22 0.98] as chosen by SNR criteria for 90% media noise 90% position jitter erf channel. 163
6.9 Spectrum of received signal equalised to target [1.00 0.87 0.15 -0.07] as chosen by MMSE criteria for 90% media noise 90% position jitter erf channel. 163
6.10 Comparison of performance for 90% media noise 90% position jitter erf channel. 164
6.11 Spectrum of unequalised received signal for 90% media noise 90% position jitter tanh channel. 164

6.12 Spectrum of received signal equalised to target [1.00 2.19 2.18 0.98] as chosen by SNR criteria for 90% media noise 90% position jitter tanh channel. 165

6.13 Spectrum of received signal equalised to target [1.00 0.81 0.09 -0.03] as chosen by MMSE criteria for 90% media noise 90% position jitter tanh channel. 165

6.14 4 state trellis showing maximum likelihood path and error event. 171

6.15 Estimated BER for white noise channel. 180

6.16 Estimated BER for jitter dominated channel with erf isolated transition. 181

6.17 Spectrum of received signal equalised to target [1.00 2.06 1.13 -0.41] as chosen by BER criteria for 70% position jitter erf channel. 182

6.18 Spectrum of received signal equalised to target [1.00 1.11 0.12 -0.14] as chosen by BER criteria for 90% media noise 90% position jitter erf channel. 182

6.19 Comparison of performance for 90% media noise 90% position jitter erf channel. 183

6.20 Spectrum of received signal equalised to target [1.00 2.97 1.13 -0.52] as chosen by BER criteria for 90% media noise 90% position jitter tanh channel. 184

6.21 Comparison of performance for 90% media noise 90% position jitter tanh channel. 184

7.1 Ripple carry adder. 188

7.2 27-bit Radix-3 Ladner-Fischer Adder. 195

7.3 27-bit Radix-3 Kogge-Stone Adder. 195

7.4 27-bit Radix-3 Generalised Ling Carry. 214

7.5 64-bit Radix-4 Generalised Ling Carry. 218

8.1 Loop elimination. 221

8.2 Double detector replaces noise predictive loop with survivor information from pre-detector. 224

8.3 Comparison of white noise MMSE detector, data dependent auto-regressive detector, data dependent noise predictive detector and double detector. 225

8.4 Estimated BER for white noise channel. 227

8.5 Comparison of performance for 90% media noise 90% position jitter erf channel. 228

List of Tables

2.1 Complexity of trellis unrolling. 43

2.2 Complexity of loop elimination for $K = 2$. 56

2.3 Complexity of loop elimination for $K = 4$. 56

2.4 Complexity comparison of trellis unrolling against loop elimination. 57

2.5 Number of operations required to sum branch metrics. 60

2.6 Path traversed by loop. 65

2.7 Transitions along each side of loop. 65

2.8 Path traversed by each loop. 67

2.9 Transitions along each side of each loop. 68

2.10 Path traversed by each loop. 70

2.11 Transitions along each side of each loop. 71

2.12 Path traversed by each loop. 72

2.13 Transitions along each side of each loop. 73

2.14 Complexity of 2T implementation. 77

2.15 Complexity of 2T standard implementation and 3T implementation with loop elimination. 78

2.16 Complexity of various 4-state implementations. 82

6.1 Comparison of performance at 12.5 dB for [1 3 3 1] AWGN channel. 160

6.2 Comparison of performance at 14.0 dB for 70% position jitter erf channel. 162

6.3 Comparison of performance at 14.0 dB for 90% media noise 90% position jitter tanh channel. 166

6.4 Comparison of performance at 14.0 dB for 70% position jitter erf channel. 182

7.1 Complexity of parallel prefix adders. 218

7.2 Complexity of Ling adders. Logic simplification of the first level is shown. Subsequent levels have the same structure as parallel prefix. 219

7.3 Complexity of generalised Ling adders. 219

8.1 Complexity comparison of trellis unrolling against loop elimination. 221

8.2 Complexity of various 4-state implementations. 223

List of Abbreviations

ACS	Add compare select unit
AWGN	Additive white Gaussian noise
BER	Bit error rate
BM	Branch metric
BMU	Branch metric unit
CSA	Compare select add unit
dB	Decibels
DDNP	Data dependent noise predictive
ISI	Intersymbol interference
LSB	Least significant bit
MMSE	Minimum mean squared error
MSB	Most significant bit
NRZ	Non return to zero
PM	Path metric
PMU	Path metric unit
PRML	Partial response maximum likelihood
RLL	Run length limited
SNR	Signal to noise ratio
SVD	Singular value decomposition
TBU	Traceback unit
VLSI	Very large scale integration

Acknowledgements

I would like to express my sincere gratitude to my academic supervisor, Dr. Oleg Zaboronski, for his friendly encouragement and invaluable guidance during the course of my research. His enthusiastic approach has made this a truly pleasurable experience.

Special thanks to the Royal Commission for the Exhibition of 1851 for the Industrial Fellowship which gave me the opportunity to continue my education whilst working in industry.

I would also like to thank all my colleagues at Arithmatica, in particular Dr. Sunil Talwar, who encouraged me early on as my Royal Commission industrial supervisor, as well as Peter Meulemans, Sam Gratrix, Nick Atkinson, Peter Collings and Tom Parnell, for their support and advice over the years.

Finally, I would like to thank my family for all their support and encouragement, in particular my parents Anne and Ray Jackson, and sister Kathy Jackson.

Declaration

I declare that, except where acknowledged, the material contained in this thesis is my own work and has not been submitted elsewhere for the purpose of obtaining an academic degree.

Robert Charles Jackson

December 2008

Abstract

As the throughput and density requirements increase for perpendicular magnetic recording channels, the presence of strong media noise degrades performance.

Detection algorithms have been developed that increase performance in channels with strong media noise through the use of data dependent detectors.

However optimal data dependent detectors are exponentially more complex than data independent detectors, and therefore cannot be fully exploited.

In this thesis we shall discuss the existing detection algorithms, comparing the performance against the complexity.

We then introduce a new sub-optimal detection algorithm, which employs a simple pre-detector that supplies estimates to a main detector. Numerical simulations are performed which show near optimal performance, but without the exponential increase in complexity.

We will also show how detector implementations can exploit structure in the trellis to further reduce complexity, through loops and path invariants.

An analytical means of measuring bit error rate from only the statistics of noise is presented, and this is utilised to optimally determine the equaliser and ISI target coefficients for a white noise Viterbi detector.

Finally, we introduce a new class of VLSI binary addition algorithms which can be utilised to increase the throughput of a Viterbi detector, but which also has a wider application in hardware design.

Summary

The first chapter *Viterbi Fundamentals* is an introduction to maximum likelihood detection and the Viterbi algorithm. We re-derive the expression for determining the sequence transmitted with maximum likelihood given the channel model and received signal, and describe how maximum likelihood detection can be visualised on a trellis. We illustrate this by consideration of the AWGN channel.

We begin the second chapter *High Throughput Viterbi Detectors* by considering existing methods for improving the throughput of Viterbi detectors. In particular, we discuss trellis unrolling, ACS retiming and bit level pipelining, and their effects on complexity.

We will then introduce the novel techniques of loop elimination and invariants, analyse their effects on complexity, and consider the reduction in complexity for practical examples.

The third chapter *High Throughput Viterbi Decoders* applies the novel principles of loop elimination and invariants to Viterbi detectors for use in communications channels, and gives a practical example.

An introduction to the magnetic channel model that we shall use is subsequent chapters is given in the forth chapter, *Magnetic Channel*.

The fifth chapter *Data Dependent Detectors* begins by introducing existing techniques for improving performance in perpendicular channels, including

auto-regressive & block diagonal data dependent detectors, and data dependent noise predictive detectors. We compare the performance of each method, and discuss the practical drawbacks of implementations of such detectors.

A novel implementation of double detector is then introduced. We compare the performance and implementation to existing methods, and discuss how the double detector achieve high performance at low complexity.

In the sixth chapter *Cost Function*, a novel approach to analytically determine the bit error rate of a Viterbi detector from only the statistics of the channel is described.

This method is then applied to the problem of determining ISI target and equaliser coefficients that minimise bit error rate, and the results are compared to existing approaches.

The final chapter *Binary Addition* introduces a novel implementation of binary addition, and compares the complexity to existing state of the art adder implementations.

Chapter 1

Viterbi Fundamentals

1.1 Introduction

Magnetic recording systems store data on a magnetic medium, so that the information can be retrieved at some point in the future.

When the information is recovered, it must be close to error free. Hard disk drives require error rates of 10^{-12} or better.

To achieve low error rates, data can be written at a low density such that each transition written on the magnetic medium results in a strong voltage from the read head which is localised to that particular transition, so the information can be recovered by peak detection.

However, there is also the underlying goal of storing as much information as possible on the magnetic medium, which is achieved by increasing the density.

But when the pulse width becomes comparable to the channel bit period, the isolated transitions overlap causing intersymbol interference (ISI) and peak detection becomes unreliable.

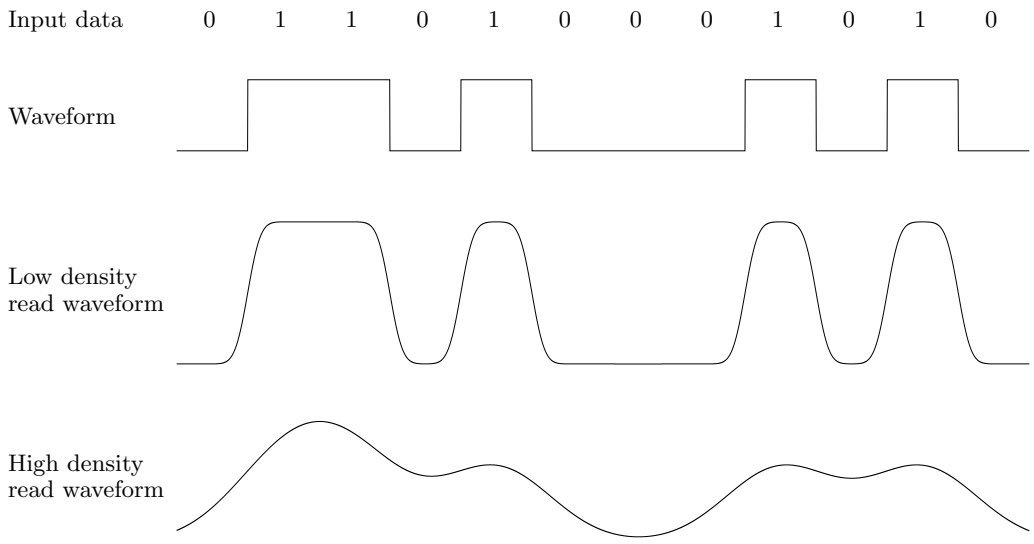


Figure 1.1: Low density waveform without ISI and high density waveform with ISI, for perpendicular magnetic recoding channel.

Partial response maximum likelihood (PRML) detection was developed as a replacement for peak detection, where the received signal is equalised to a predetermined target, and maximum likelihood decoding is used to recover the information.

The Viterbi algorithm was first introduced in 1967 [8] as a method of decoding convolutional codes. In 1972, Forney [9] showed that the Viterbi algorithm solves the maximum likelihood sequence detection problem optimally in the presence of intersymbol interference and additive white noise.

Kobayashi and Tang [10] were the first to apply the Viterbi algorithm to magnetic recoding, and PRML subsequently became firmly established by

read channel manufacturers.

As density continues to increase and greater throughput is required, the effects of random changes in position and phase of the isolated transitions become increasingly important. These effects are characterised by correlations between noise samples. Kavcic [11] introduced signal dependent autoregressive Viterbi detectors which take such correlations into account.

Note also that the Viterbi algorithm has many applications outside of magnetic recoding, particularly in communications systems such as CDMA and GSM digital mobile phones, dial-up modems, satellite and deep-space communications, wireless and wired networking.

1.2 Maximum Likelihood Detection

In this section, we start by introducing the maximum likelihood sequence detection problem.

We will reformulate the solution, and in the process introduce branch metrics and path metrics, then map the reformulated solution to a trellis representation.

Finally we describe the Viterbi algorithm and the windowed Viterbi algorithm.

Theorem 1.2.1 (Maximum Likelihood Sequence Detection)**.** Suppose that equiprobable binary sequences $\underline{x} \in \{0,1\}^N$ are transmitted over a noisy chan-

nel,

$$r_i = f(r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) \tag{1.2.1}$$

Then given the received sequence $\underline{r} \in \mathbb{R}^N$, the original unencoded sequence $\hat{\underline{x}} \in \{0,1\}^N$ which maximises the conditional probability density,

$$\hat{\underline{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmax}} \{\rho(\underline{x} \mid \underline{r})\} \tag{1.2.2}$$

is obtained by maximising the following function

$$\hat{\underline{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmax}} \left\{ \prod_{i=0}^{N-1} \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) \right\} \tag{1.2.3}$$

Proof. Applying Bayes' theorem [12] to (1.2.2) yields,

$$\hat{\underline{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmax}} \left\{ \frac{\rho(\underline{r} \mid \underline{x})\rho(\underline{x})}{\rho(\underline{r})} \right\} \tag{1.2.4}$$

Since all input sequences $\underline{x} \in \{0,1\}^N$ are assumed to be equiprobable, and as $\rho(\underline{r})$ is independent of the maximisation argument $\underline{x}$, we have the following,

$$\hat{\underline{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmax}} \{\rho(\underline{r} \mid \underline{x})\} \tag{1.2.5}$$

The sequence $\hat{\underline{x}}$ is the maximally likely estimate of sequence $\underline{x}$. The conditional probability density $\rho(\underline{r} \mid \underline{x})$ is called the maximal likelihood function (as a function of $\underline{x}$).

Using the joint probability density identity,

$$\rho(A, B \mid C) = \rho(A \mid B, C)\rho(B \mid C) \tag{1.2.6}$$

the maximal likelihood function can be expanded using repeated application of (1.2.6),

$$\rho(\underline{r} \mid \underline{x}) = \prod_{i=0}^{N-1} \rho(r_i \mid r_{i-1}, \ldots, r_0, \underline{x}) \tag{1.2.7}$$

Moreover, because of the assumed causality of the signal, the received signal r_i only depends on $x_i, \ldots, x_0$ from the original sequence.

Note that in general, transitions on both sides of the sampling point affect the signal. If however this influence is finite, one can enumerate received signals in such a way that the channel looks causal.

Therefore,

$$\rho(\underline{r} \mid \underline{x}) = \prod_{i=0}^{N-1} \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) \tag{1.2.8}$$

□

Corollary 1.2.2. Maximising (1.2.3) is equivalent to the following minimisation

$$\underline{\hat{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} -\ln \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) \right\} \tag{1.2.9}$$

Proof. The natural logarithm function is strictly increasing and therefore has no effect on the argument chosen in the maximisation. The negation simply changes the problem from maximisation to minimisation. □

Definition 1.2.3 (Branch Metric & Path Metric)**.** We define the following function of the probability density to be the branch metric at time i,

$$BM^{(i)}(x_i, \ldots, x_0) = -\ln \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) \tag{1.2.10}$$

Note that for brevity, we have omitted the dependence on the received signal from the notation.

Therefore we can express the maximum likelihood function in (1.2.9) as

$$\underline{\hat{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} BM^{(i)}(x_i, \ldots, x_0) \right\} \tag{1.2.11}$$

Note that since probability densities take values in the range $[0, 1]$, the branch metric is non-negative.

The path metric at time t for path $x_t, \ldots, x_0$ is defined to be

$$PM^{(t)}(x_t, \ldots, x_0) = \sum_{i=0}^{t} BM^{(i)}(x_i, \ldots, x_0) \tag{1.2.12}$$

Therefore we can express the maximum likelihood function in (1.2.9) as

$$\underline{\hat{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmin}} \left\{ PM^{(N-1)}(\underline{r}, \underline{x}) \right\} \tag{1.2.13}$$

Lemma 1.2.4. For a given path $\underline{x}$, the path metric at time t can be determined from the path metric at time $t-1$ and the branch metric at time

t.

$$PM^{(t)}(x_t, \ldots, x_0) = BM^{(t)}(x_t, \ldots, x_0) + PM^{(t-1)}(x_{t-1}, \ldots, x_0) \quad (1.2.14)$$

Proof.

$$\begin{aligned} PM^{(t)}(x_t, \ldots, x_0) &= \sum_{i=0}^{t} BM^{(i)}(x_i, \ldots, x_0) \\ &= BM^{(t)}(x_t, \ldots, x_0) + \sum_{i=0}^{t-1} BM^{(i)}(x_i, \ldots, x_0) \\ &= BM^{(t)}(x_t, \ldots, x_0) + PM^{(t-1)}(x_{t-1}, \ldots, x_0) \end{aligned} \quad (1.2.15)$$

□

Definition 1.2.5 (Markov Channel)**.** A Markov channel is a binary input channel with the following property

$$\rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) = \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_{i-K+1}) \quad (1.2.16)$$

where $K > 0$ is the constraint length.

Corollary 1.2.6. Suppose a binary sequence $\underline{x} \in \{0,1\}^N$ is transmitted over a Markov channel with constraint length K. Then the maximum likelihood function is given by

$$\hat{\underline{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmax}} \left\{ \prod_{i=0}^{N-1} \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_{i-K+1}) \right\} \quad (1.2.17)$$

or equivalently

$$\underline{\hat{x}} = \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} BM^{(i)}(x_i, \ldots, x_{i-K+1}) \right\} \tag{1.2.18}$$

Proof. Apply the Markov channel property (1.2.16) to the maximum likelihood function in (1.2.3) and (1.2.9). □

By direct inspection, the maximum likelihood path is found by comparing the received signal $\underline{r}$ to all 2^N possible ideal signals corresponding to the 2^N possible input sequences $\underline{x}$.

However, we can use a dynamic programming algorithm called the Viterbi algorithm.

Definition 1.2.7. Define the surviving path metric for all paths ending with the sequence $x_t, \ldots, x_{t-K+2}$ to be

$$SPM^{(t)}(x_t, \ldots, x_{t-K+2}) = \min_{x_{t-K+1},\ldots,x_0} PM^{(t)}(x_t, \ldots, x_0) \tag{1.2.19}$$

Theorem 1.2.8. The surviving path metric for all paths ending with the sequence $x_t, \ldots, x_{t-K+2}$ can be calculated recursively as

$$\begin{aligned} &SPM^{(t)}(x_t, \ldots, x_{t-K+2}) \\ &\quad = \min_{x_{t-K+1}\in\{0,1\}} \left\{ BM^t(x_t, \ldots, x_{t-K+1}) + SPM^{(t-1)}(x_{t-1}, \ldots, x_{t-K+1}) \right\} \end{aligned} \tag{1.2.20}$$

Proof. From the definition in (1.2.19), separate x_{t-K+1} from the minimisation. Then apply the path metric decomposition from (1.2.14), and factor

the branch metric from the inner minimisation.

$$
\begin{aligned}
&SPM^{(t)}(x_t, \ldots, x_{t-K+2}) \\
&\quad = \min_{x_{t-K+1}, \ldots, x_0} PM^{(t)}(x_t, \ldots, x_0) \\
&\quad = \min_{x_{t-K+1} \in \{0,1\}} \left\{ \min_{x_{t-K}, \ldots, x_0} PM^{(t)}(x_t, \ldots, x_0) \right\} \\
&\quad = \min_{x_{t-K+1} \in \{0,1\}} \left\{ \min_{x_{t-K}, \ldots, x_0} \left\{ BM^{(t)}(x_t, \ldots, x_{t-K+1}) + PM^{(t-1)}(x_{t-1}, \ldots, x_0) \right\} \right\} \\
&\quad = \min_{x_{t-K+1} \in \{0,1\}} \left\{ BM^{(t)}(x_t, \ldots, x_{t-K+1}) + \min_{x_{t-K}, \ldots, x_0} PM^{(t-1)}(x_{t-1}, \ldots, x_0) \right\} \\
&\quad = \min_{x_{t-K+1} \in \{0,1\}} \left\{ BM^{(t)}(x_t, \ldots, x_{t-K+1}) + SPM^{(t-1)}(x_{t-1}, \ldots, x_{t-K+1}) \right\}
\end{aligned}
\tag{1.2.21}
$$

□

Corollary 1.2.9. The path metric of the maximum likelihood path,

$$
PM(\underline{\hat{x}}) = \min_{\underline{x} \in \{0,1\}^N} PM^{(N-1)}(x_{N-1}, \ldots, x_0) \tag{1.2.22}
$$

can be obtained by minimising over all surviving path metrics ending in $x_{N-1}, \ldots, x_{N-K+1}$

$$
PM(\underline{\hat{x}}) = \min_{x_{N-1}, \ldots, x_{N-K+1}} SPM^{(N-1)}(x_{N-1}, \ldots, x_{N-K+1}) \tag{1.2.23}
$$

Proof. Separate the minimisation into two stages, then substitute using the

definition in (1.2.19)

$$\begin{aligned} PM(\hat{\underline{x}}) &= \min_{\underline{x}\in\{0,1\}^N} PM^{(N-1)}(x_{N-1},\ldots,x_0) \\ &= \min_{x_{N-1},\ldots,x_{N-K+1}} \left\{ \min_{x_{N-K},\ldots,x_0} PM^{(N-1)}(x_{N-1},\ldots,x_0) \right\} \qquad (1.2.24) \\ &= \min_{x_{N-1},\ldots,x_{N-K+1}} SPM^{(N-1)}(x_{N-1},\ldots,x_{N-K+1}) \end{aligned}$$

□

The above allows us to visualise the maximum likelihood path on a trellis. The trellis can be described as follows

- The trellis has length N.
- At each time t, there are 2^{K-1} states labelled $x_{t-K+2},\ldots,x_t$.
- Each state $x_{t-K+2},\ldots,x_t$ is connected to two previous states, namely $0,x_{t-K+2},\ldots,x_{t-1}$ and $1,x_{t-K+2},\ldots,x_{t-1}$.
- Each state $x_{t-K+2},\ldots,x_t$ is connected to two subsequent states, namely $x_{t-K+2},\ldots,x_t,0$ and $x_{t-K+2},\ldots,x_t,1$.
- The state $x_{t-K+2},\ldots,x_t$ holds the value of the surviving path metric $SPM^{(t)}(x_t,\ldots,x_{t-K+2})$.
- The edge connecting states $x_{t-K+1},\ldots,x_{t-1}$ and $x_{t-K+2},\ldots,x_t$ contains the branch metric $BM^{(t)}(x_t,\ldots,x_{t-K+1})$.

By construction, the trellis has the following properties

- Every sequence $x_0,\ldots,x_{N-1}$ is represented uniquely as a path through the trellis.

- For a given path through the trellis, the sum of the edges traversed by the path is equal to the path metric.
- The maximum likelihood path corresponds to a unique path through the trellis.

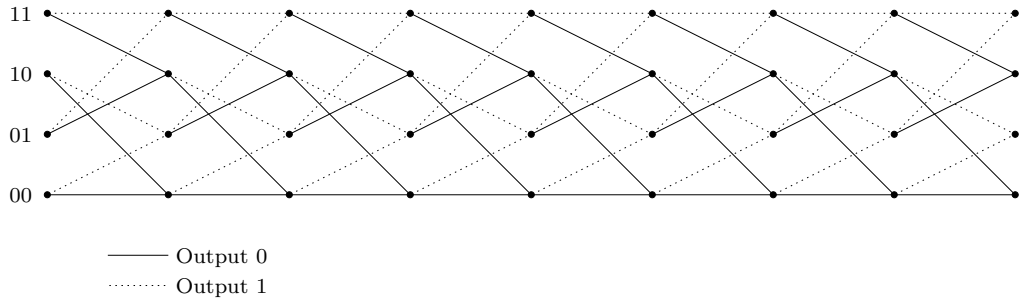


Figure 1.2: 4 state trellis.

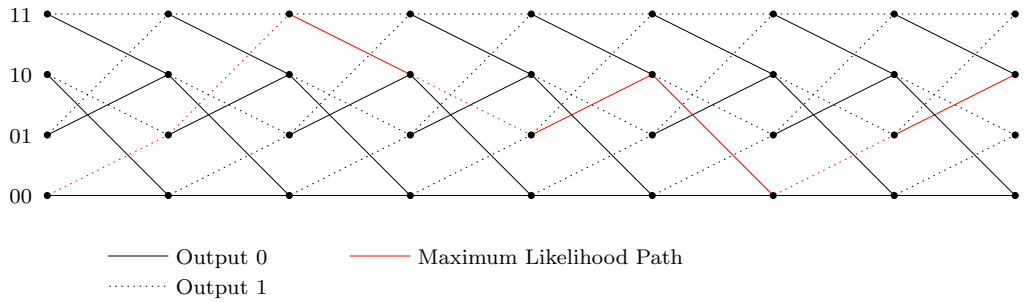


Figure 1.3: 4 state trellis showing maximum likelihood path.

Corollary 1.2.10. The surviving path metric at state $x_{t-K+2}, \ldots, x_t$ is found on the trellis by summing the surviving path metric at each connected previous state with the corresponding branch metric, and minimising over each state.

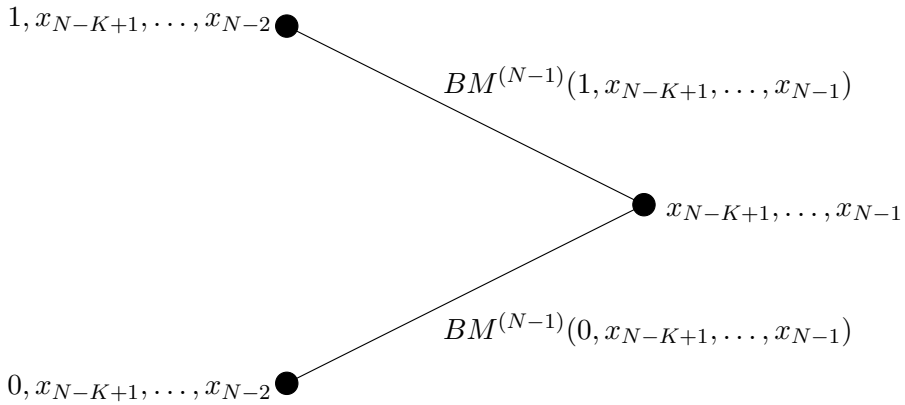


Figure 1.4: Two branches joining at a single state.

Proof. From (1.2.20) we have the following

$$\begin{aligned} &SPM^{(t)}(x_t, \ldots, x_{t-K+2}) \\ &\quad = \min_{x_{t-K+1} \in \{0,1\}} \left\{ BM^{(t)}(x_t, \ldots, x_{t-K+1}) + SPM^{(t-1)}(x_{t-1}, \ldots, x_{t-K+1}) \right\} \end{aligned} \tag{1.2.25}$$

By construction, state $x_{t-K+2}, \ldots, x_t$ is connected to states $0, x_{t-K+2}, \ldots, x_{t-1}$ and $1, x_{t-K+2}, \ldots, x_{t-1}$. The edges connecting state $x_{t-K+1}, \ldots, x_{t-1}$ to state $x_{t-K+2}, \ldots, x_t$ contains the branch metric $BM^{(t)}(x_t, \ldots, x_{t-K+1})$ and the previous states contain the surviving path metrics $SPM^{(t-1)}(x_{t-1}, \ldots, x_{t-K+1})$.

□

Notation. Henceforth, we denote the surviving path metric at state i as $PM_i^{(t)}$, and the branch metric connecting states j and i as $BM_{j,i}^{(t)}$.

Algorithm 1.2.11 (Viterbi Algorithm for Markov channel)**.** The following describes the Viterbi algorithm [8, 9].

1. for $i = 0$ to $2^{K-1} - 1$

1.1. initialise $PM_i^{(0)} = 0$

2. for $t = 1$ to N

2.1. for $i = 0$ to $2^{K-1} - 1$

2.1.1. let j, k be the previous states connected to current state i

2.1.2. then $j = \left\lfloor \frac{1}{2} i \right\rfloor$ and $k = \left\lfloor \frac{1}{2} i \right\rfloor + 2^{K-2}$

2.1.3. $PM_i^{(t)} = \min \left\{ PM_j^{(t-1)} + BM_{j,i}^{(t)}, PM_k^{(t-1)} + BM_{k,i}^{(t)} \right\}$

2.1.4. store the decision $\operatorname{argmin} \left\{ PM_j^{(t-1)} + BM_{j,i}^{(t)}, PM_k^{(t-1)} + BM_{k,i}^{(t)} \right\}$

3. determine the final state, f, which contains the smallest path metric $f = \operatorname{argmin}_i PM_i^{(N)}$

4. from the final state f, traceback the maximum likelihood path through the trellis following the stored decisions

5. output the maximum likelihood path

Theorem 1.2.12 (Viterbi Algorithm for Markov channel)**.** The Viterbi algorithm in 1.2.11 determines the maximum likelihood path through the trellis.

For proof, refer to [9].

The Viterbi algorithm is not restricted to such regular trellises, and can be generalised to any finite state trellis.

However, for the remainder of this thesis, we shall restrict ourselves to considerations of regular trellises, but it should be noted that many of the results extend to an arbitrary trellis.

Example 1.2.13 ((1,4) RLL code)**.** Consider the trellis for a (1,4) RLL code, where the minimum length of a run is 1, and the maximum length of a run

is 4, where a *run* refers to consecutive inputs with the same value. Encode the states by the length of the current run $\{1, 2, 3, 4\}$, and the value of each term in the current run $\{0, 1\}$.

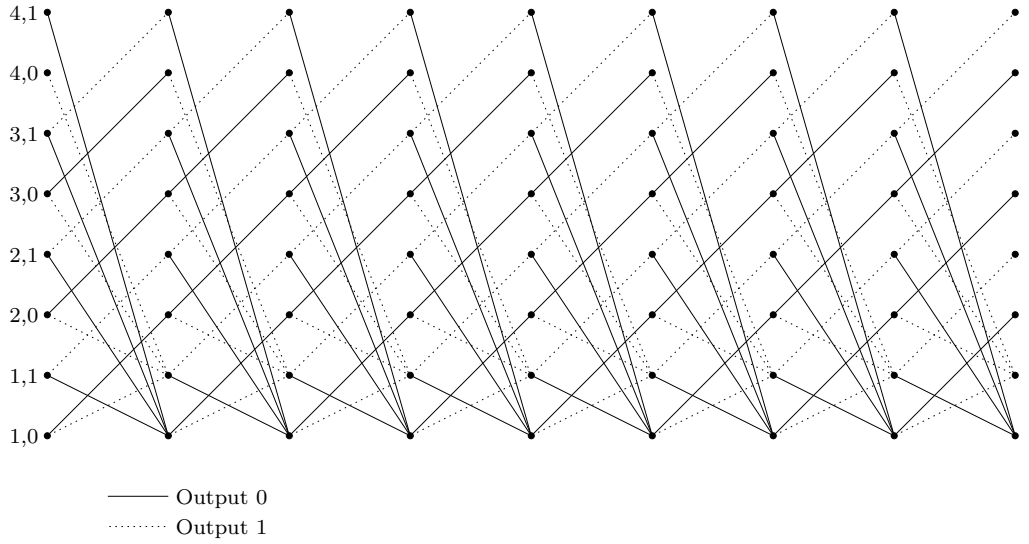


Figure 1.5: 8 state (1,4) RLL trellis.

Algorithm 1.2.14 (Viterbi Algorithm for any finite state trellis)**.** The following described the Viterbi Algorithm for any finite state trellis.

1. foreach state i

1.1. initialise $PM_i^{(0)} = 0$

2. for $t = 1$ to N

2.1. foreach state i

2.1.1. let S_i be the set of states at time $t-1$ connected to state i at time t

2.1.2. $PM_i^{(t)} = \min_{j \in S_i} \left\{ PM_j^{(t-1)} + BM_{j,i}^{(t)} \right\}$

2.1.3. store the decision $\operatorname{argmin}_{j \in S_i} \left\{ PM_j^{(t-1)} + BM_{j,i}^{(t)} \right\}$

3. determine the final state, f, which contains the smallest path metric $f = \text{argmin}_i \, PM_i^{(N)}$

4. from the final state f, traceback the maximum likelihood path through the trellis following the stored decisions

5. output the maximum likelihood path

Theorem 1.2.15 (Viterbi Algorithm for any finite state trellis)**.** The Viterbi algorithm in 1.2.14 determines the maximum likelihood path through the trellis.

For proof, refer to [9].

The Viterbi algorithm proceeds in two discrete steps. Firstly we move forward through the trellis eliminating contending paths, leaving only a single surviving path for each state and storing the decision we made as to which path survives. Having proceeded forward through the trellis, we choose the minimum of the surviving path metrics at the final state, and trace the decisions backwards through the trellis.

We have so far restricted our considerations to finite length trellises, but we can further generalise the Viterbi algorithm to infinite trellises, by using a sliding window approach together with the observation that all contending paths converge with high probability after relatively few time steps ($5K$ and $10K$ are common estimates for the convergence length).

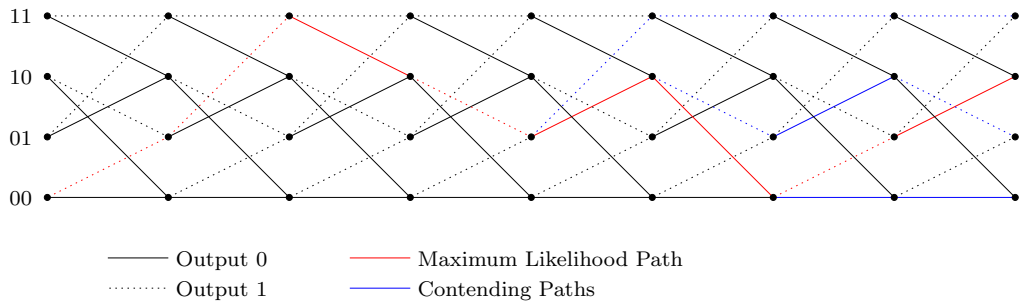


Figure 1.6: 4 state trellis showing maximum likelihood path and convergence of contending paths.

Algorithm 1.2.16 (Windowed Viterbi Algorithm)**.** The Viterbi algorithm determines the maximum likelihood path through the trellis as follows

1. foreach state i

1.1. initialise $PM_i^{(0)} = 0$

2. for $t > 0$

2.1. foreach state i

2.1.1. let S_i be the set of states at time $t-1$ connected to state i at time t

2.1.2. $PM_i^{(t)} = \min_{j \in S_i} \left\{ PM_j^{(t-1)} + BM_{j,i}^{(t)} \right\}$

2.1.3. store the decision $\operatorname{argmin}_{j \in S_i} \left\{ PM_j^{(t-1)} + BM_{j,i}^{(t)} \right\}$

2.1.4. determine the current state with the smallest path metric $s_{min} = \operatorname{argmin}_i PM_i^{(t)}$

2.1.5. from the state s_{min}, traceback the maximum likelihood path back T time steps through the trellis following the stored decisions

2.1.6. output the decision at time $t - T$

where T is the traceback length.

In 2.1.5. we traceback from the best current state and hence this is referred to as best state traceback. But since all contending paths are assumed to eventually converge, we may choose a fixed state (usually the zero state) to traceback from. This is known as zero state traceback. Note that a longer traceback length is required for zero state traceback to achieve similar performance to best state traceback.

Theorem 1.2.17 (Windowed Viterbi Algorithm)**.** The windowed Viterbi algorithm as stated in 1.2.16 approaches the performance of the Viterbi algorithm as the traceback length is increased.

For proof, refer to [9].

1.3 White Noise Viterbi Detector

We can now solve the maximum likelihood sequence detection problem in the presence of intersymbol interference and additive white noise [9].

Theorem 1.3.1. Suppose a binary sequence $\underline{x} \in \{0,1\}^N$ which is encoded with a non-recursive convolutional code describing inter-symbol interference (ISI) with impulse response $\{g_0, \ldots, g_I\}$,

$$y_i = \sum_{k=0}^{I} g_k x_{i-k} \tag{1.3.1}$$

is transmitted over an additive white Gaussian noise (AWGN) channel,

$$r_i = y_i + \omega_i \tag{1.3.2}$$

where $\omega_i \sim N(0, \sigma^2)$. Then given the received sequence $\underline{r} \in \mathbb{R}^N$, the original unencoded sequence $\underline{\hat{x}} \in \{0,1\}^N$ transmitted with maximum likelihood is given by

$$\underline{\hat{x}} = \underset{\underline{x} \in \{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} \left(r_i - \sum_{k=0}^{I} g_k x_{i-k} \right)^2 \right\} \tag{1.3.3}$$

Proof. Since $\omega_i \sim N(0, \sigma^2)$, the noise component of the received signals are uncorrelated, therefore,

$$\begin{aligned} \rho(\underline{r} \mid \underline{x}) &= \prod_{i=0}^{N-1} \rho(r_i \mid r_{i-1}, \ldots, r_0, x_i, \ldots, x_0) \\ &= \prod_{i=0}^{N-1} \rho(r_i \mid x_i, \ldots, x_0) \\ &= \prod_{i=0}^{N-1} \rho(r_i \mid x_i, \ldots, x_{i-I}) \end{aligned} \tag{1.3.4}$$

Furthermore, the probability density $\rho(r_i \mid x_i, \ldots, x_{i-I})$ is given by,

$$\rho(r_i \mid x_i, \ldots, x_{i-I}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \omega_i^2 \right) \tag{1.3.5}$$

which using (1.3.2) can be expressed as,

$$\rho(r_i \mid x_i, \ldots, x_{i-I}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \left(r_i - \sum_{k=0}^{I} g_k x_{i-k} \right)^2 \right) \tag{1.3.6}$$

Therefore by (1.2.3), the maximally likely sequence $\hat{\underline{x}}$ is obtained by,

$$\begin{aligned}\hat{\underline{x}} &= \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmax}} \left\{ \prod_{i=0}^{N-1} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \left(r_i - \sum_{k=0}^{I} g_k x_{i-k} \right)^2 \right) \right\} \\ &= \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \prod_{i=0}^{N-1} \exp\left(r_i - \sum_{k=0}^{I} g_k x_{i-k} \right)^2 \right\} \\ &= \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} \left(r_i - \sum_{k=0}^{I} g_k x_{i-k} \right)^2 \right\}\end{aligned} \tag{1.3.7}$$

□

Definition 1.3.2 (White Noise Branch Metric)**.** The white noise branch metric is given by

$$BM^{(i)}(r_i, x_{i-I}, \ldots, x_i) = \left(r_i - \sum_{k=0}^{I} g_k x_{i-k} \right)^2 \tag{1.3.8}$$

Note that in the above branch metric, the term r_i^2 is common to all branches at i and can therefore be subtracted from each branch as the argument of the minimisation in (1.2.3) will not be effected.

Therefore we can equivalently use the following definition for white noise branch metric, which does not require the square of the received signal.

Definition 1.3.3 (Simplified White Noise Branch Metric)**.** The simplified white noise branch metric is given by

$$BM^{(i)}(r_i, x_{i-I}, \ldots, x_i) = \left(\sum_{k=0}^{I} g_k x_{i-k} \right)^2 - 2r_i \left(\sum_{k=0}^{I} g_k x_{i-k} \right) \tag{1.3.9}$$

Chapter 2

High Throughput Viterbi Detectors

In this chapter, we will investigate how to increase the throughput of Viterbi detectors without prohibitively increasing the complexity.

Firstly we shall discuss loop elimination, which improves throughput of high radix Viterbi detectors at the expense an exponential increase in complexity. Then we shall discuss invariants of initial and final states which lowers complexity by exploiting the properties of path differences between the sides of loops, and finally demonstrate loop elimination and invariants with an implementation example.

2.1 Trellis Unrolling

Implementations of the Viterbi algorithm can be separated into three distinct parts.

First the branch metric unit (BMU) takes the received signal and produces all the required branch metrics and passed them to the path metric unit (PMU). The PMU adds the branch metrics to the accumulated path metrics and discards all but the shortest path to each state. This is referred to as add-compare-select (ACS) and the PMU consists of an ACS unit for each state. The decisions made by the ACS units are passed to the traceback unit (TBU) which keeps track of each path so the shortest path can eventually be traced back from the final state.

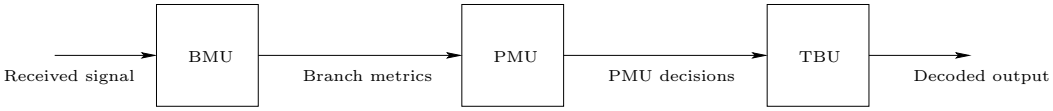


Figure 2.1: Viterbi detector implementation comprising branch metric, path metric and traceback units.

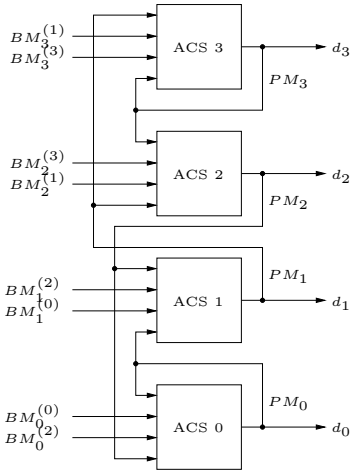


Figure 2.2: Path metric unit comprising ACS units for each state.

The ACS units inside the PMU are connected by a feedback loop, since the previous path metrics are required in order to compute the new path metric.

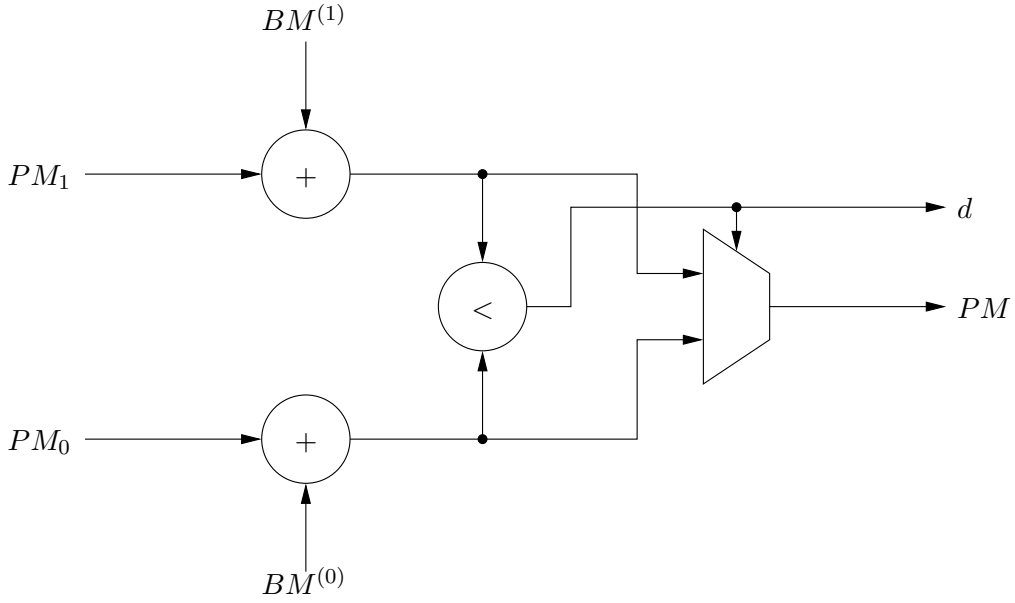


Figure 2.3: 1T ACS unit.

As shown in figure 2.3, a 1T ACS unit comprises 2 two input adders which sum the previous path metrics with the branch metrics, and a single two input minimiser to select the smaller of the two new contending path metrics.

In a hardware implementation, the adders may be implemented in parallel, therefore the critical path through the ACS unit runs through one of the two input adders, then through the two input minimiser.

The delay and cell area for a two input adder and two input minimiser are comparable, therefore we shall consider them both to have unit delay and area. Hence a 1T ACS unit has an area of 3 units and a delay of 2 units.

It is possible to accelerate the ACS computation using carrysave arithmetic

for the addition [13], which allows one of the adders to be replaced by a 4:2 compressor. It should be noted that at the small bit widths generally used in Viterbi detectors, typically less than 8-bits, this saving will be negligible and comes at the expense of increased complexity, since the non-carrysave sums containing the contending path metrics must also be computed before the surviving path metric is selected by the result of the comparison.

To improve the throughput of the Viterbi algorithm, a 1T trellis, such as figure 1.2, may be unrolled into the 2T trellis of figure 2.4.

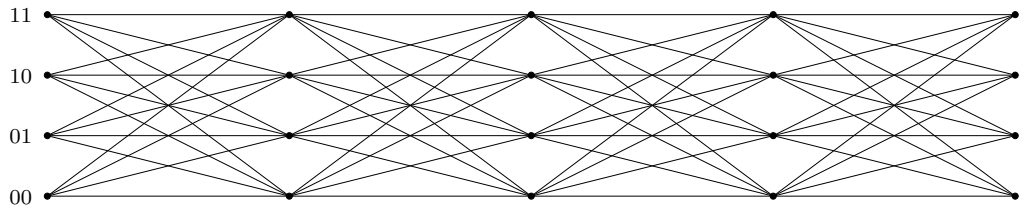


Figure 2.4: 4 state 2T trellis.

In the regular 1T trellis, two path are compared at each state, with the longer path being discarded in favour of the shorter path.

For the 2T trellis, two steps from the 1T trellis are combined together. We must therefore compare four paths at each state, discarding the three longest paths in favour of the shortest path.

As shown in figure 2.5, a 2T ACS unit comprises 4 two input adders which sum the previous path metrics with the branch metrics, and a 3 two input minimiser to select the smallest of the four new contending path metrics. Unrolling the ACS recursion to perform two trellis iterations in a single cycles has been used to improve throughput in [14, 15].

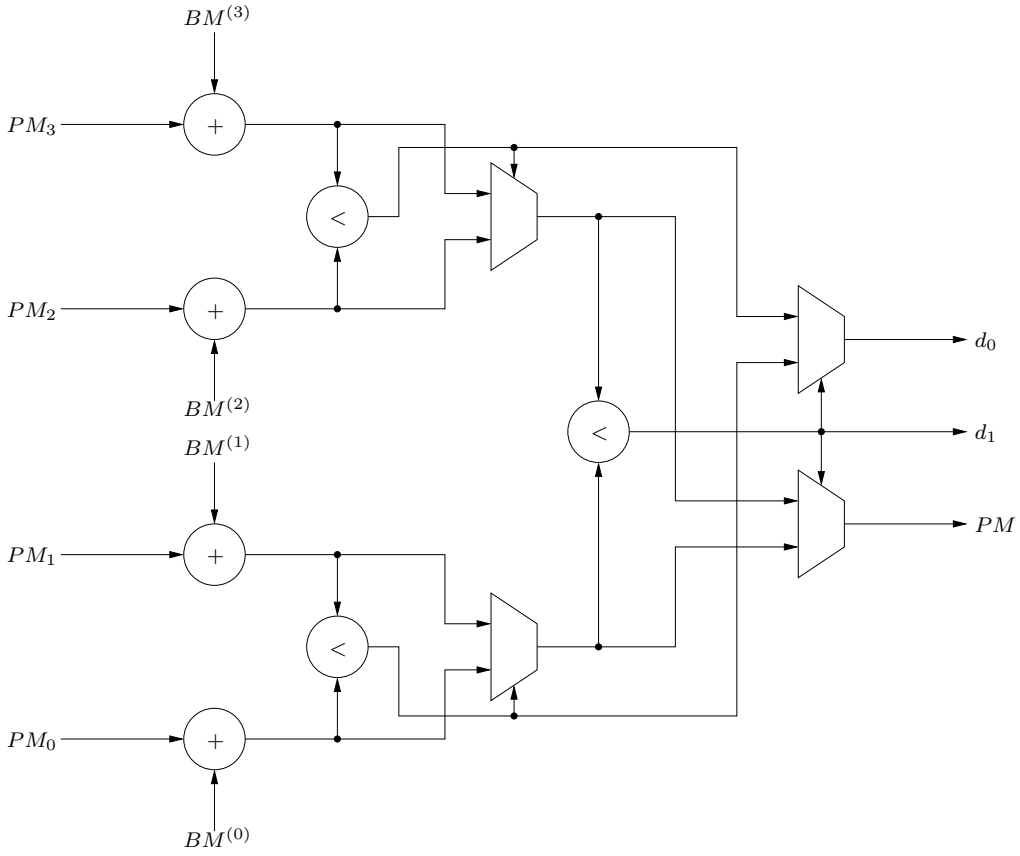


Figure 2.5: 2T ACS unit.

This can be generalised to a T-T trellis, where T steps from the 1T trellis are combined together such that 2^T paths converge at each state. For example $2^3 = 8$ paths must be compared in the 3T trellis shown in figure 2.6. A

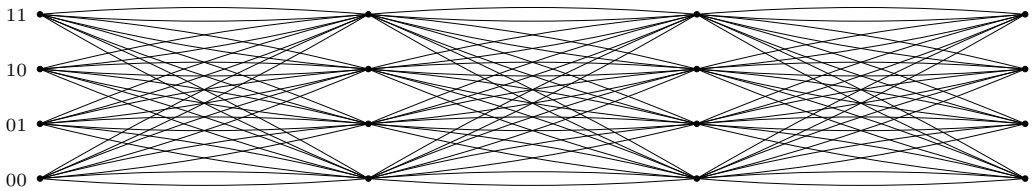


Figure 2.6: 4 state 3T trellis.

T-T ACS unit comprises 2^T two input adders which sum the previous path metrics with the branch metrics, and a $2^T - 1$ two input minimiser to select

the smallest of the four new contending path metrics.

Unrolling the trellis in this manner is intended to to improve throughput, at the expense of complexity. But it should be noted that continued unrolling achieves diminishing improvements in throughput, at the expense of exponentially increasing area.

In particular, a single T-T ACS unit has a delay of $T+1$ units and an area of $2^{(T+1)}-1$ units, since the 2^T adders are completely in parallel, and the 2^T-1 comparators are constructed in a logarithmic tree.

Since a T-T ACS unit produces T outputs per cycles, whilst a 1T ACS unit only produces a single output per cycle, the delay metric of importance is delay per output. Therefore a 1T ACS unit has a delay per output of 2 unit, whilst a T-T ACS unit has a delay per output of $(T+1)/T$ units.

Below is a table which summarises the complexity for T-T implementations.

T-T	Delay	Delay per output	ACS area
1	2	2.00	3
2	3	1.50	7
3	4	1.33	15
4	5	1.25	31
T	$T+1$	$(T+1)/T$	$2^{T+1}-1$

Table 2.1: Complexity of trellis unrolling.

Table 2.1 shows that a 4T implementation, which is less than twice as fast as a 1T implementation, has an area which is over 10 times greater. This increase in area precludes the practical use of unrolled implementations. In reality, designs over 2T are rarely used as the diminishing returns cannot be justified.

2.2 ACS Retiming

An improvement of the standard ACS unit can be achieved by retiming the operations. Figure 2.7 shows how the operations are regrouped into retimed cycles.

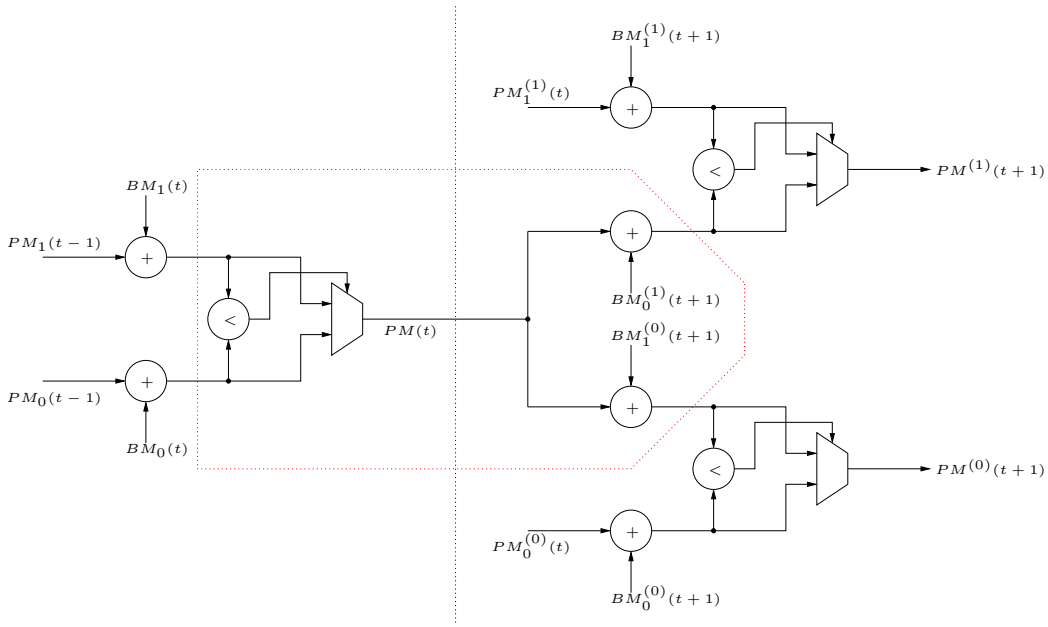


Figure 2.7: Two iterations of 1T ACS unit, with 1T CSA retiming region indicated.

Having retimed the ACS unit, we obtained a modified unit in which the order of operations is compare-select-add, hence we refer to this retimed structure as a CSA unit.

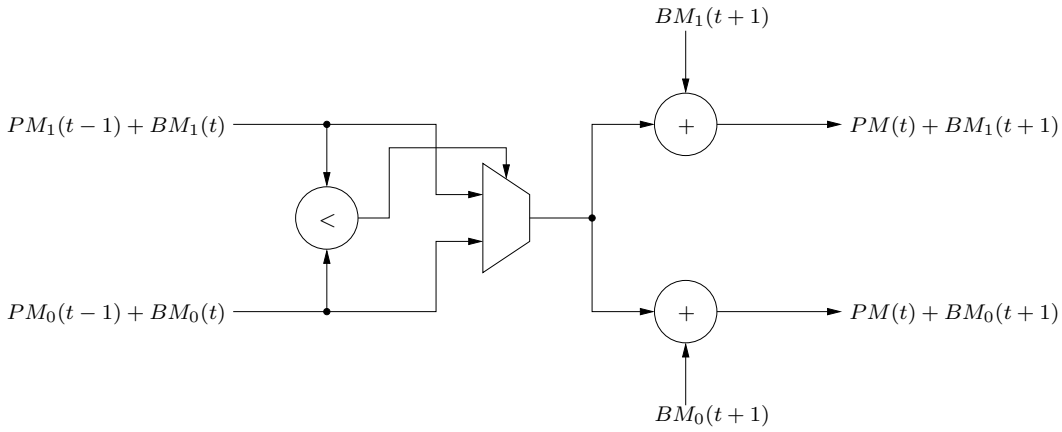


Figure 2.8: Unoptimised 1T CSA unit.

Then by reordering the multiplexer and adders, which results in twice as many adders, we obtain the optimised CSA architecture where the branch metric addition is performed in parallel with the comparison [16, 17].

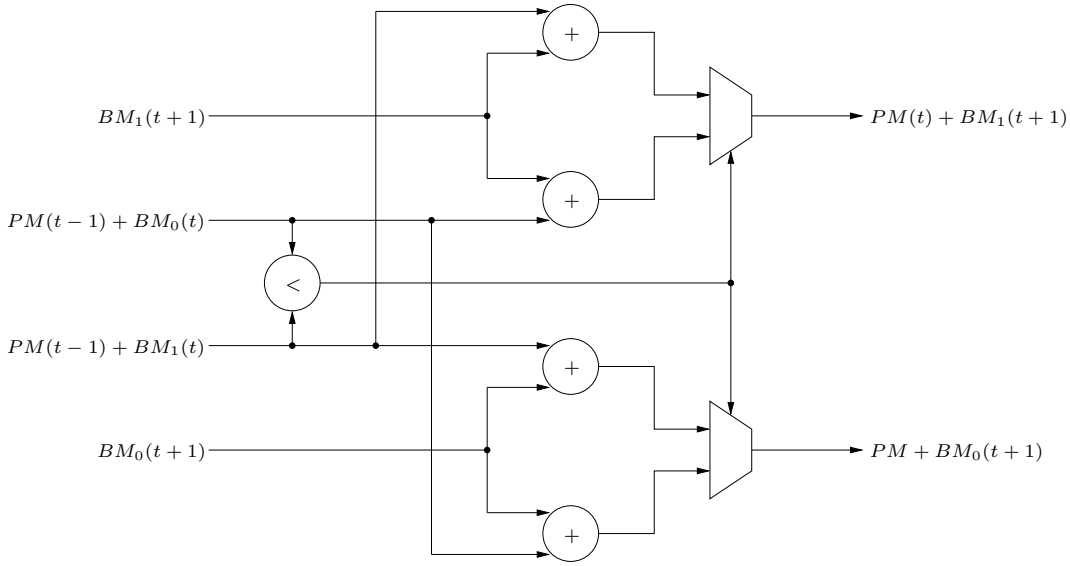


Figure 2.9: Optimised 1T CSA unit.

The result is an architecture where the critical path consists of a single adder,

followed by a multiplexer, thereby eliminating an entire comparator from the critical path.

Note that in the higher radix generalisation of the CSA unit, the branch metric addition can be performed in parallel with the comparison of the incoming path metrics (with pre-added branch metrics). However the comparison will be between 2^T terms for a T-T CSA unit, which requires T two input comparators. Therefore the delay per output remains constant as radix increases, hence performance cannot be improved significantly by using high radix architectures.

2.3 ACS Bit Level Pipelining

The conventional ACS unit cannot be pipelined due to the dependence of the inputs of the current cycle, to the outputs of the previous cycle.

Bit-level pipelining utilises a redundant number system [18] and carrysave addition [19–21] in which the ACS operation can be expressed in such a way that carries only propagate to the next bit position. A practical implementation was shown in [15].

The critical operation is reduced to the time taken to compute a single bit slice which is constant regardless of the bit width of the path metrics. However, the small bit widths of typical path metrics, the redundant number system and carrysave representation make the implementation of the single slice sufficiently complex to yield little improvement over CSA schemes.

2.4 Loop Elimination

In this section, we introduce the principle of loop elimination [3], which aims to increase the throughput of Viterbi detectors further than unrolling, and to achieve this without an exponential increase in complexity.

Consider all paths which converge at a given final state of a K time step section of a 1T trellis, where K is the constraint length. For example, consider all paths converging at state 1 of 3 time slices of a 4 state trellis, as shown in figure 2.10

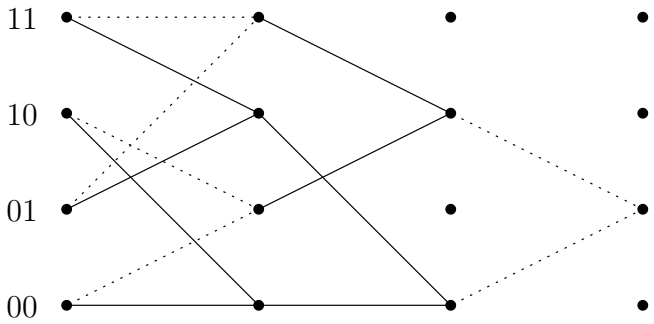


Figure 2.10: All paths converging at state 1 of 3 time slices of a 4 state trellis.

Note that there are exactly two paths connecting each initial state to each final state, as illustrated in our example 2.11.

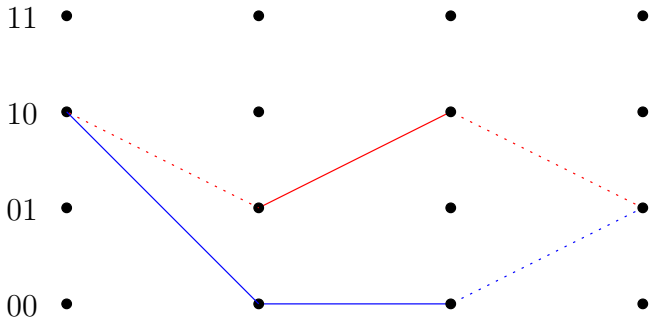


Figure 2.11: Exactly two paths connecting initial and final state.

Definition 2.4.1 (Loop)**.** We define a *loop* to be the portion of two distinct paths between an initial state and a final state through which both paths pass. The two *sides* of the loop correspond to the route through the trellis traversed by the two distinct paths. The *length* of the loop is defined to be the number of time steps between the initial and final states.

In figure 2.11, we illustrate a loop of length 3, from initial state 2, to final state 1. The two sides of the loop are shown in blue and red.

Let us consider the 8 contending paths being considered at the final state of our example. These 8 paths can be characterised by the states the pass through. In particular, the paths are $\{0,0,0,1\}$, $\{2,0,0,1\}$, $\{1,2,0,1\}$, $\{3,2,0,1\}$, $\{0,1,2,1\}$, $\{2,1,2,1\}$, $\{1,3,2,1\}$ and $\{3,3,2,1\}$.

Therefore the surviving path metric is determined by

$$\begin{aligned} PM_1^{(t)} = \min \Big\{ & PM_0^{(t-3)} + BM_{0,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & PM_2^{(t-3)} + BM_{2,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & PM_1^{(t-3)} + BM_{1,2}^{(t-2)} + BM_{2,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & PM_3^{(t-3)} + BM_{3,2}^{(t-2)} + BM_{2,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & PM_0^{(t-3)} + BM_{0,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)}, \\ & PM_2^{(t-3)} + BM_{2,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)}, \\ & PM_1^{(t-3)} + BM_{1,3}^{(t-2)} + BM_{3,2}^{(t-1)} + BM_{2,0}^{(t)}, \\ & PM_3^{(t-3)} + BM_{3,3}^{(t-2)} + BM_{3,2}^{(t-1)} + BM_{2,0}^{(t)} \Big\} \end{aligned} \tag{2.4.1}$$

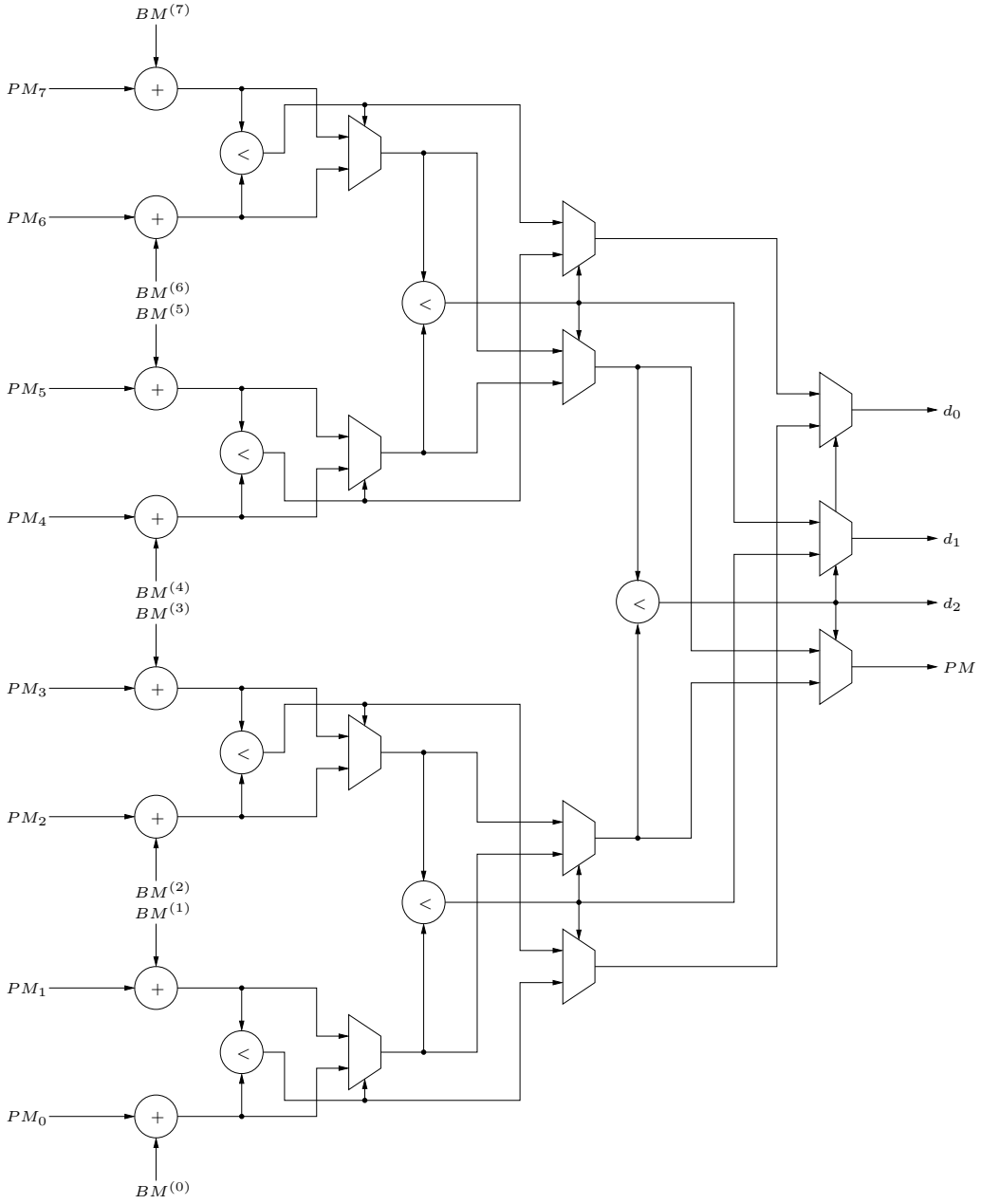


Figure 2.12: 3T ACS unit.

Note that we can collect together terms which originate at the same initial

state, and factor the path metric to those initial states.

$$\begin{aligned} PM_1^{(t)} = \min\Big\{ & PM_0^{(t-3)} + \min\Big\{ BM_{0,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & \qquad BM_{0,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)} \Big\}, \\ & PM_1^{(t-3)} + \min\Big\{ BM_{1,2}^{(t-2)} + BM_{2,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & \qquad BM_{1,3}^{(t-2)} + BM_{3,2}^{(t-1)} + BM_{2,0}^{(t)} \Big\}, \\ & PM_2^{(t-3)} + \min\Big\{ BM_{2,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & \qquad BM_{2,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)} \Big\}, \\ & PM_3^{(t-3)} + \min\Big\{ BM_{3,2}^{(t-2)} + BM_{2,0}^{(t-1)} + BM_{0,0}^{(t)}, \\ & \qquad BM_{3,3}^{(t-2)} + BM_{3,2}^{(t-1)} + BM_{2,0}^{(t)} \Big\}, \Big\} \end{aligned} \tag{2.4.2}$$

The collected terms correspond to loops in the trellis, with the minimisation selecting the shortest side of the loop and eliminating the longest side of the loop. We refer to the selection of the shortest side of a loop as *loop elimination*.

For example $\min\left\{ BM_{2,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, BM_{2,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)} \right\}$ corresponds to the loop shown in figure 2.11.

Notice that loop elimination is independent of the global path metric, and depends only on the branch metrics along the loop itself. This is advantageous for hardware implementations as loops can be eliminated outside of the critical path metric feedback loop, resulting in a minimisation of fewer terms and consequently a shorter critical path.

For example, let BM_i be the combined branch metric for the surviving side

of the loop from initial state i.

$$\begin{aligned}
BM_0 &= \min\left\{BM_{0,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, BM_{0,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)}\right\} \\
BM_1 &= \min\left\{BM_{1,2}^{(t-2)} + BM_{2,0}^{(t-1)} + BM_{0,0}^{(t)}, BM_{1,3}^{(t-2)} + BM_{3,2}^{(t-1)} + BM_{2,0}^{(t)}\right\} \\
BM_2 &= \min\left\{BM_{2,0}^{(t-2)} + BM_{0,0}^{(t-1)} + BM_{0,0}^{(t)}, BM_{2,1}^{(t-2)} + BM_{1,2}^{(t-1)} + BM_{2,0}^{(t)}\right\} \\
BM_3 &= \min\left\{BM_{3,2}^{(t-2)} + BM_{2,0}^{(t-1)} + BM_{0,0}^{(t)}, BM_{3,3}^{(t-2)} + BM_{3,2}^{(t-1)} + BM_{2,0}^{(t)}\right\}
\end{aligned} \tag{2.4.3}$$

Then the path metric is calculated as the minimisation over four terms (identical to a 2T ACS unit), rather than eight (for the unmodified 3T ACS unit).

$$PM_1^{(t)} = \min_{i\in\{0,1,2,3\}}\left\{PM_i^{(t-3)} + BM_i\right\} \tag{2.4.4}$$

Consequently, the delay through the 3T ACS unit is the same as through a 2T ACS unit.

The unmodified 3T ACS unit is shown in figure 2.13. Note than inside each red box, the same path metric enters both additions and can therefore be push through the minimiser. This transformation results in figure 2.14, which is identical to the 2T ACS unit shown in figure 2.5 if the initial branch metric minimisations are precomputed.

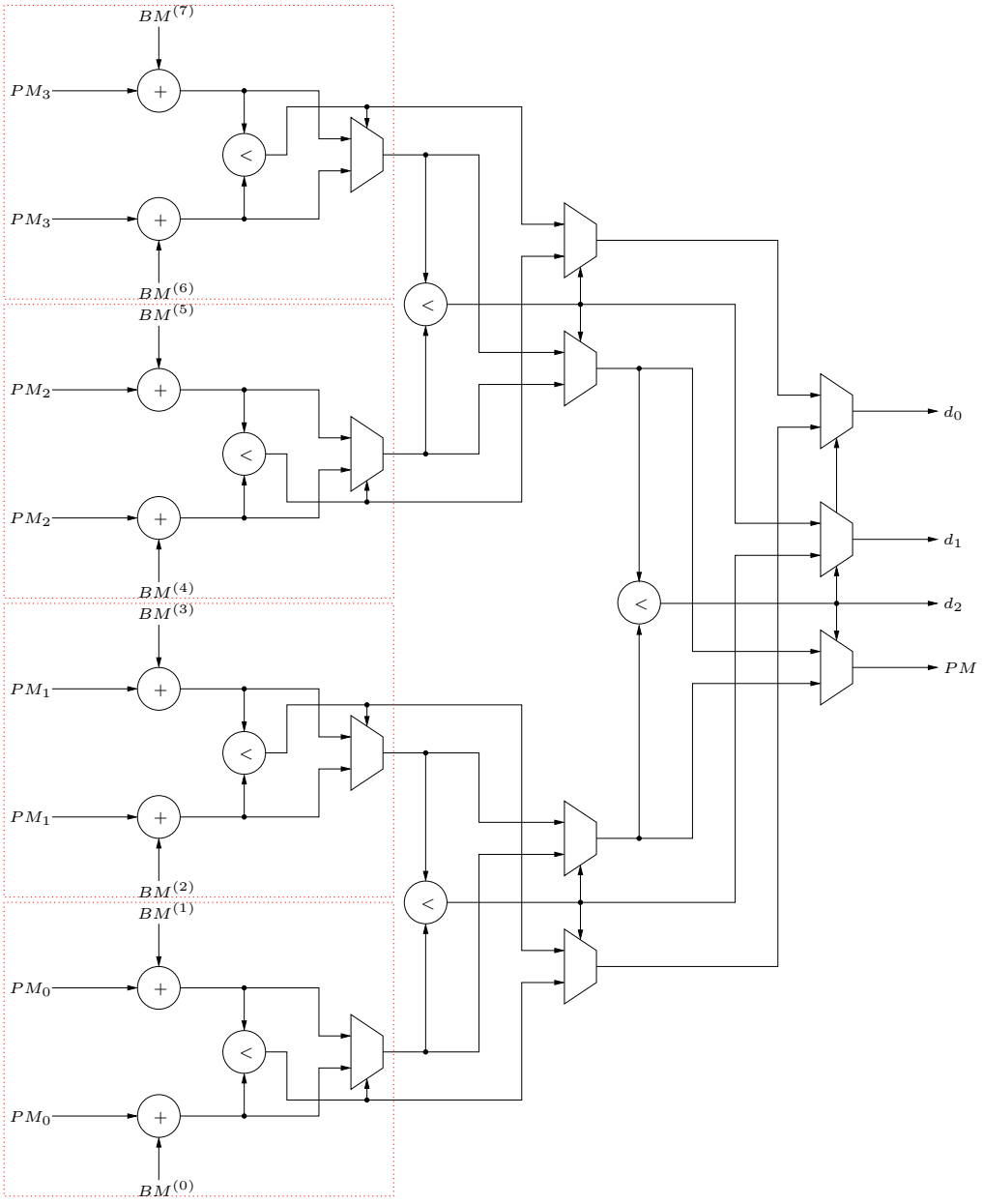


Figure 2.13: 3T ACS unit with duplicated input path metrics. Red boxes indicate region to be transformed.

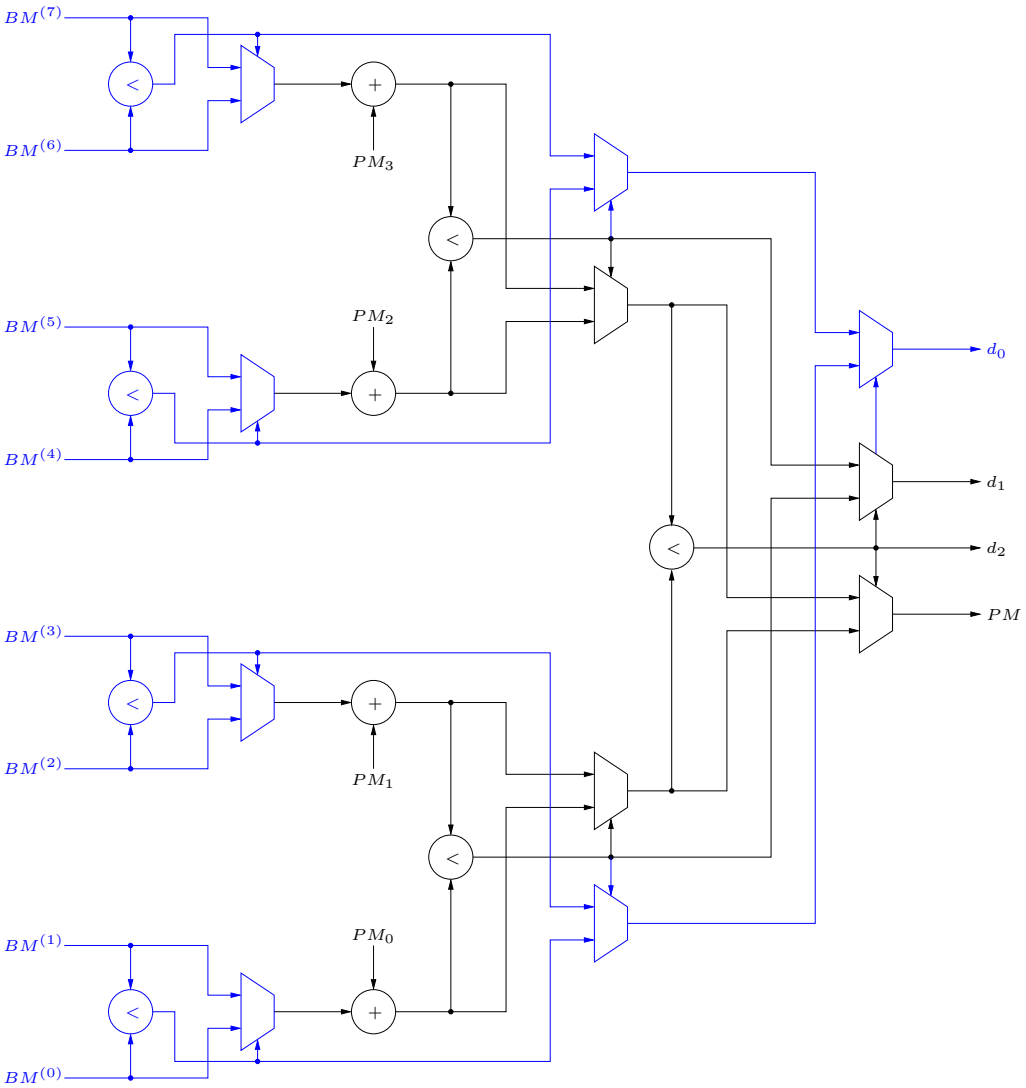


Figure 2.14: 3T ACS unit with duplicated input path metrics, transformed to a 2T ACS unit (shown in black).

Figure 2.15 shows the above example of loop elimination on the trellis. The loops are clearly evident on the 3T trellis in figure 2.15(b), and having eliminated the loops, the trellis degenerated to a 2T trellis as shown in figure 2.15(e).

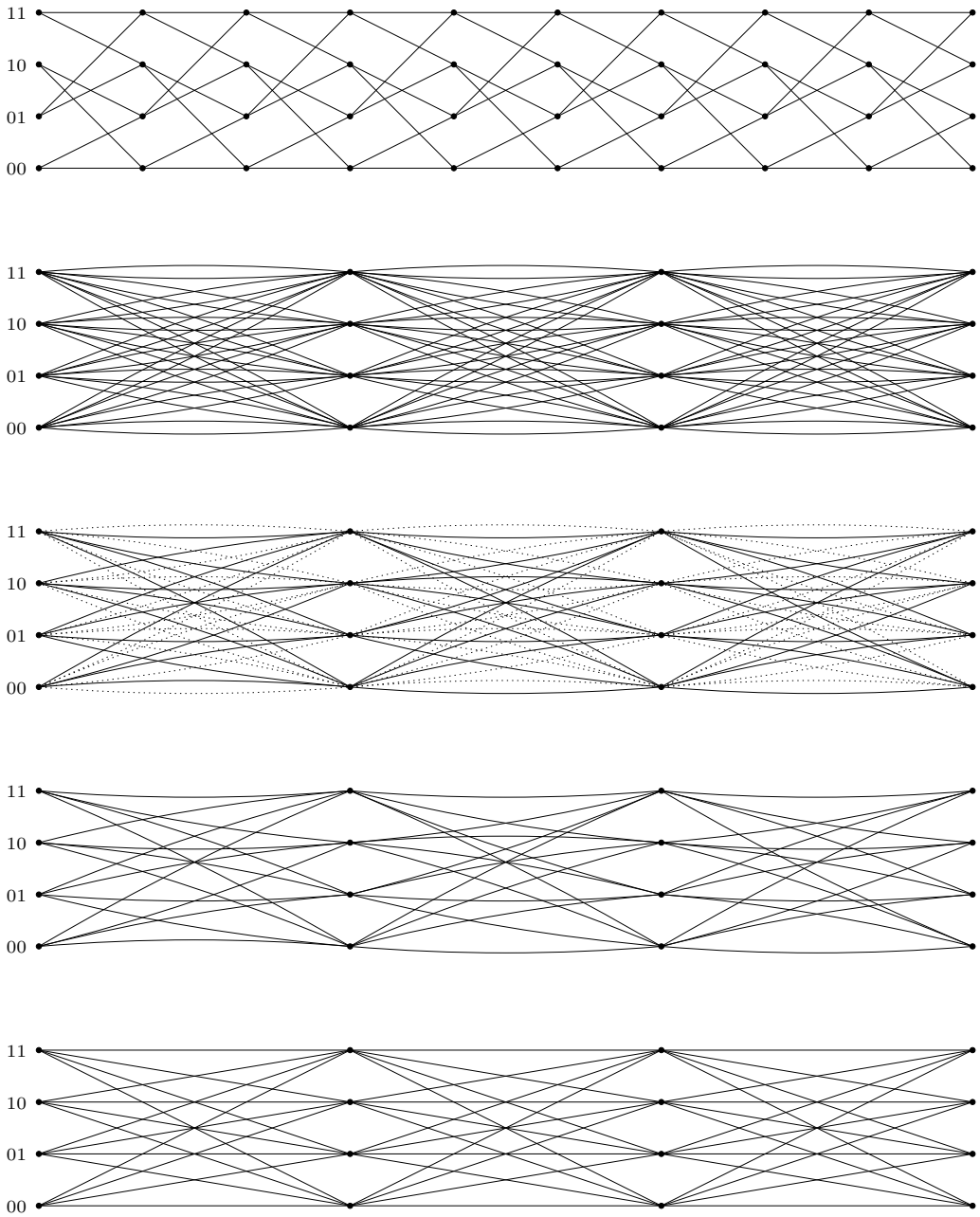


Figure 2.15: 4 state trellis illustrating loop elimination. (a) shows original 1T trellis. (b) equivalent 3T trellis. (c) compare sides of each loop. (d) remove longest side of each loop. (e) 3T trellis with loops eliminated.

Loop elimination can be repeatedly applied. For example, consider a the two

state trellis in figure 2.16(a).

First eliminate the loops in each 2T trellis section. This results in the trellis shown in figure 2.16(b). Note that there are now loops of length 4 which can be eliminated, resulting in trellis figure 2.16(c), which in turn has loops of length 8.

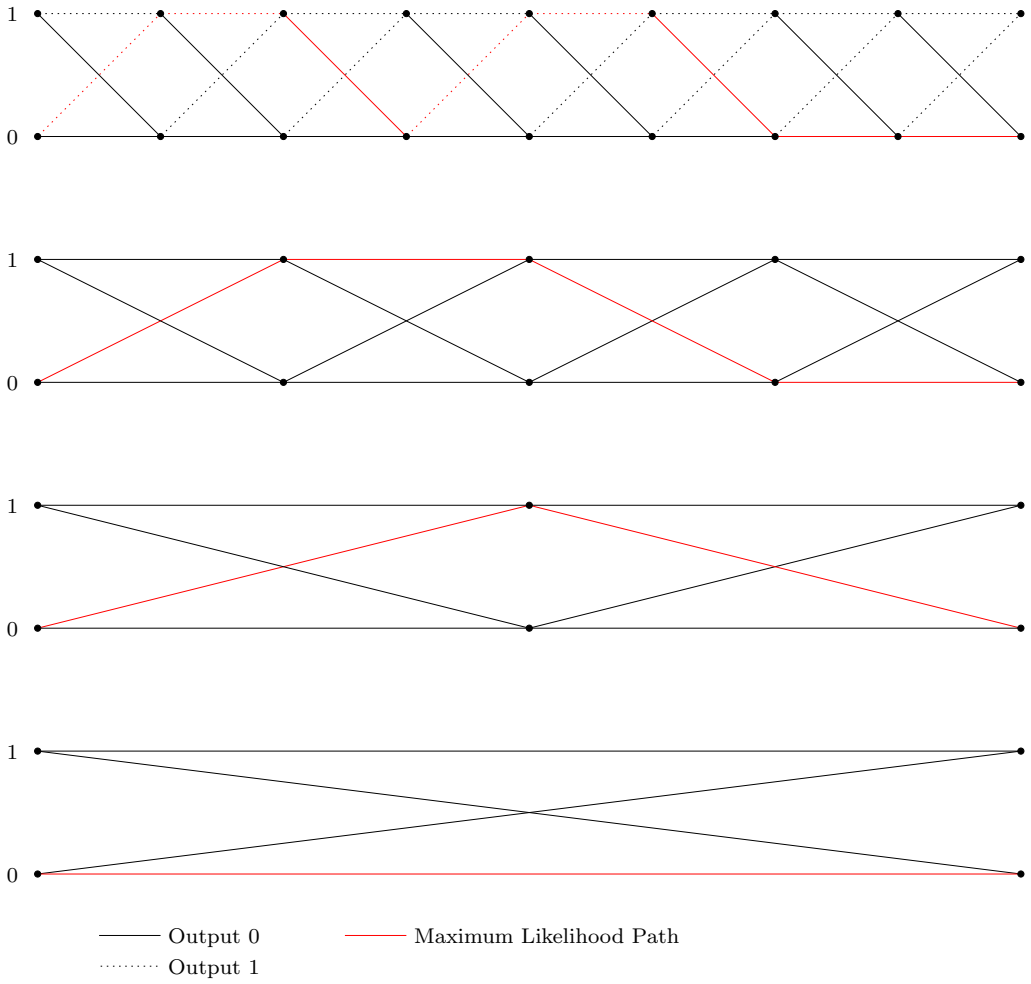


Figure 2.16: 2 state trellis illustrating recursive loop elimination.

This results in a highly parallel Viterbi implementation, which in hardware has a delay proportional to the logarithm of the block length.

2.4.1 Loop Elimination Complexity

Since the shortest loops occur in a K-T trellis section, where K is the constraint length, we can only utilise loop elimination for T-T trellis sections when $T \geq K$.

Therefore we analyse two examples when $K = 2$ and $K = 4$, before analysing the general case.

Below is a table comparing the complexity for high radix implementations using loop elimination for constraint length $K = 2$.

T	Delay	Delay/output	ACS area	Loop elim. area	Total area
1	2	2.00	3	0	3
2	2	1.00	3	2	5
3	2	0.67	3	6	9
4	2	0.50	3	14	17
T	2	$2/T$	3	$2(2^{T-1}-1)$	2^T+1

Table 2.2: Complexity of loop elimination for $K = 2$.

Note that when $K = 2$, the delay through an ACS unit is always equivalent to that of a 1T ACS unit.

Below is a table comparing the complexity for high radix implementations using loop elimination for constraint length $K = 4$.

T	Delay	Delay/output	ACS area	Loop elim. area	Total area
1	2	2.00	3	0	3
2	3	1.50	7	0	7
3	4	1.33	15	0	15
4	4	1.00	15	8	23
T	4	$4/T$	15	$8(2^{T-3}-1)$	2^T+7

Table 2.3: Complexity of loop elimination for $K = 4$.

Note that when $K = 4$, the delay through an ACS unit never exceeds that of a 3T ACS unit.

Let us now consider the general case of a constraint length K trellis, and compare loop elimination against the standard loop unrolling algorithm.

Below is a table comparing the complexity for high radix implementations using trellis unrolling and loop elimination for constraint length K.

Method	Delay	Delay/output	ACS area	Loop elimination area	Total area
Trellis unrolling	$T+1$	$(T+1)/T$	$2^{T+1}-1$	0	$2^{T+1}-1$
Loop elimination	K	K/T	$2^{K}-1$	$2^{K-1}(2^{T-K+1}-1)$	$2^{T}+2^{K-1}-1$

Table 2.4: Complexity comparison of trellis unrolling against loop elimination.

Note that with loop elimination, the delay through an ACS unit never exceeds that of a K-T ACS unit, but the delay increases linearly when using trellis unrolling.

However the total area for both implementations increases exponentially with T. In the next section we show how the complexity of loop elimination can be reduced in order to reduce the area.

2.5 Asymptotic Complexity of PMU

In this section we determine a bound for the complexity of the path metric unit comprising the ACS unit and the arithmetic required to sum the 1T branch metrics, but not including the calculation of the 1T branch metrics themselves.

We shall consider two implementations of a K-T Viterbi detector for a convolutional code with constraint length K. The first implementation is a 1T implementation unrolled T times, whilst the second is a true K-T implementation.

The advantage of the K-T implementation over the unrolled 1T implementation is the critical feedback loop only contains 1 addition and K minimisations, whilst the unrolled 1T implementation contains both K additions and minimisations.

Therefore the K-T implementation is faster, but it is useful to understand the trade-off between performance and complexity.

2.5.1 Complexity of Unrolled 1T Implementation

For each of the 2^{K-1} states at each of the K time slices, we need to compute

$$P = \min(P_0 + B_0, P_1 + B_1) \tag{2.5.1}$$

which requires three operations. Therefore we require a total of

$$C_{1T} = 3K2^{K-1} \tag{2.5.2}$$

operations, or

$$\mathcal{C}_{1T} = 3K \tag{2.5.3}$$

operations per state.

2.5.2 Complexity of K-T Implementation

Each of the final 2^{K-1} states is connected by exactly two paths to each initial state. We can compute the new path metric by summing the branch metrics along each path, then eliminate the loop to leave exactly one path from each initial state, finally minimising between the 2^{K-1} remaining paths. We shall refer to the operation count for each of the above as C_B, C_L and C_M respectfully.

The final minimisation for each state between the 2^{K-1} remaining paths requires 2^{K-1} additions to add the accumulated branch metrics to the old path metrics, and $2^{K-1} - 1$ minimisations to find the minimum of the 2^{K-1} contenders. This requires a total of

$$C_M = 2^{2K-1} - 2^{K-1} \tag{2.5.4}$$

operations.

We also need to eliminate the loops from each initial state to each final state by minimising between the sum of the branch metrics along each path. Therefore this requires

$$C_L = 2^{2K-2} \tag{2.5.5}$$

minimisations.

In order to calculate the sum of the branch metrics along each path in the minimum number of operations, we form the additions using a binary tree. For simplicity, assume that $K = 2^k$ is a power of two.

At the first level of the tree, we separate the radix-2 trellis into $K/2$ 2T sections. Each intermediate state has two paths entering and two paths leaving, therefore we require 4 additions per state to form each new radix-4 section of the trellis. We now have a radix-4 trellis of length $K/2$-T.

Similarly at the second level of the tree, we separate the radix-4 trellis into $K/4$ 2T sections. Each intermediate state has four paths entering and four paths leaving, therefore we require 16 additions per state to form each new radix-16 section of the trellis. We now have a radix-16 trellis of length $K/4$-T.

We continue this process until at the final K-th level we are left with a radix $2^{2^k} = 2^K$ trellis of length 1T. To summarise the above, we repeatedly increase the radix of the trellis by combining neighbouring sections. This requires the following operations

Level	Initial Radix	Final Radix	Adders/State	Sections	Total Adders
1	2	4	4	$K/2$	$4 \cdot K/2 \cdot 2^{K-1}$
2	4	16	16	$K/4$	$16 \cdot K/4 \cdot 2^{K-1}$
3	16	256	256	$K/8$	$256 \cdot K/8 \cdot 2^{K-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	$2^{2^{i-1}}$	2^{2^i}	2^{2^i}	$K/2^i$	$2^{2^i} \cdot K/2^i \cdot 2^{K-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	$2^{2^{k-1}} = 2^{K-2}$	$2^{2^k} = 2^K$	2^K	1	$2^K \cdot 1 \cdot 2^{K-1}$

Table 2.5: Number of operations required to sum branch metrics.

Therefore the total number of additions required is

$$\begin{aligned} C_B &= K2^{K-1} \sum_{i=1}^{\log_2 K} \frac{2^{2^i}}{2^i} \\ &= K2^{K-1} \sum_{i=1}^{\log_2 K} 2^{2^i - i} \\ &= 2^{2K-1} + K2^{K-1} \sum_{i=1}^{(\log_2 K)-1} 2^{2^i - i} \\ &= 2^{2K-1} + K2^{K-1} \sum_{i=1}^{\log_2 \frac{K}{2}} 2^{2^i - i} \end{aligned} \tag{2.5.6}$$

Since $2^i - i \geq 1$ for $i \geq 1$, a lower bound for C_B is given by

$$\begin{aligned} C_B &\geq 2^{2K-1} + K2^{K-1} \sum_{i=1}^{\log_2 \frac{K}{2}} 2 \\ &= 2^{2K-1} + \left(K \log_2 \frac{K}{2} \right) 2^K \end{aligned} \tag{2.5.7}$$

The terms in (2.5.6) are monotonically increasing, therefore the each term in the sum may be upper bounded by the final term.

$$\begin{aligned} C_B &\leq 2^{2K-1} + K2^{K-1} \sum_{i=1}^{\log_2 \frac{K}{2}} 2^{2^{\log_2 \frac{K}{2}} - \log_2 \frac{K}{2}} \\ &= 2^{2K-1} + K2^{K-1} \sum_{i=1}^{\log_2 \frac{K}{2}} \frac{2}{K} 2^{\frac{K}{2}} \\ &= 2^{2K-1} + \left(\log_2 \frac{K}{2} \right) 2^{\frac{3K}{2}} \end{aligned} \tag{2.5.8}$$

Therefore the operation count to calculate the branch metrics and eliminate

the loops is bounded by

$$3 \cdot 2^{2K-2} + \left(K \log_2 \frac{K}{2} \right) 2^K \leq C_B + C_L \leq 3 \cdot 2^{2K-2} + \left(\log_2 \frac{K}{2} \right) 2^{\frac{3K}{2}} \tag{2.5.9}$$

and the total operation count

$$C = C_B + C_L + C_M \tag{2.5.10}$$

is bounded by

$$5 \cdot 2^{2K-2} - 2^{K-1} + \left(K \log_2 \frac{K}{2} \right) 2^K \leq C \leq 5 \cdot 2^{2K-2} - 2^{K-1} + \left(\log_2 \frac{K}{2} \right) 2^{\frac{3K}{2}} \tag{2.5.11}$$

Hence the number of operations per state is bounded by

$$5 \cdot 2^{K-1} - 1 + 2K \log_2 \frac{K}{2} \leq \mathcal{C} \leq 5 \cdot 2^{K-1} - 1 + \left(\log_2 \frac{K}{2} \right) 2^{\frac{K}{2}-1} \tag{2.5.12}$$

If we compare the leading order term of (2.5.3) against the leading order term of (2.5.12), we see that in order to take advantage of high radix implementations and loops using a K-T implementation, the complexity increases exponentially compared to simple unrolling whose complexity increases linearly.

However we have shown loop elimination offers a real performance advantage. In the next section we shall see how path invariants can be utilised to reduce the complexity and make high radix implementation practical to implement.

2.6 Invariants

The first invariant we consider allows us to determine which of the two contending paths that form a loop is shortest, for all of the 2^{K-1} states, by performing only K multiplications.

To determine the surviving path to a state, the branch metrics are added to the old path metrics, and the smaller of the two values is chosen.

$$PM_i^{(t)} = \min\left\{PM_0^{(t-K)} + BM_{0,i}, PM_1^{(t-K)} + BM_{1,i}\right\} \tag{2.6.1}$$

When determining the shortest side of a loop, both sides originate from the same state. Therefore $PM_0^{(t-K)} = PM_1^{(t-K)}$.

$$PM_i^{(t)} = PM_0^{(t-K)} + \min\left\{BM_{0,i}, BM_{1,i}\right\} \tag{2.6.2}$$

Therefore we must find which branch metric is the smallest. If we assume the Euclidean metric, the branch metrics are given by

$$\begin{aligned} BM_{0,i} &= \sum_{k=0}^{K-1} (r_{t-k} - a_k)^2 \\ BM_{1,i} &= \sum_{k=0}^{K-1} (r_{t-k} - b_k)^2 \end{aligned} \tag{2.6.3}$$

where $\{a_k\}$ and $\{b_k\}$ are the branch labels on the two sides of the loop.

To determine the smallest branch metric, we must compare them

$$\Delta = BM_{0,i} - BM_{1,i} \tag{2.6.4}$$

But we can simplify this as follows

$$\begin{aligned}\Delta &= BM_{0,i} - BM_{1,i} \\ &= \sum_{k=0}^{K-1} (r_{t-k} - a_k)^2 - \sum_{k=0}^{K-1} (r_{t-k} - b_k)^2 \\ &= \sum_{k=0}^{K-1} (r_{t-k} - a_k)^2 - (r_{t-k} - b_k)^2 \\ &= \sum_{k=0}^{K-1} a_k^2 - b_k^2 - 2(a_k - b_k) r_{t-k}\end{aligned} \tag{2.6.5}$$

For every loop, the branch labels $\{a_k\}$ and $\{b_k\}$ are different. But the difference between the branch labels in invariant, and as we will show below, is actually related to the generator polynomial

$$a_k - b_k = g_k \tag{2.6.6}$$

where $\{g_k\}$ are the generator polynomial coefficients. Therefore

$$\Delta = \sum_{k=0}^{K-1} a_k^2 - b_k^2 - 2 g_k r_{t-k} \tag{2.6.7}$$

The $\{a_k^2\}$ and $\{b_k^2\}$ terms are constants, which can be precomputed, and the multiplication with the received signal can be shared amounts all the loops. This greatly simplifies the loop elimination algorithm implementation, and gives us our first invariant.

Theorem 2.6.1. Given a Viterbi decoder of constraint length k, with encoding given by polynomial $g_0 + g_1 D + \ldots + g_{k-1} D^{k-1}$, then the difference between branch labels along any loop is given by $\pm g_i$ for $i = 0, \ldots, k-1$.

Proof. The encoder has 2^{k-1} states. By definition, the two sides of a loop begin and end in the same state, say initial state $\alpha_0 \dots \alpha_{k-2}$ and final state $\beta_0 \dots \beta_{k-2}$.

A path through the trellis from the initial state to the final state must pass through $k-1$ other states between:

State Number	State
Initial	$\alpha_0 \dots \alpha_{k-2}$
1	$\alpha_1 \dots \alpha_{k-2} x_0$
2	$\alpha_2 \dots \alpha_{k-2} x_0 x_1$
n	$\alpha_n \dots \alpha_{k-2} x_0 \dots x_{n-1}$
$k-2$	$\alpha_{k-1} x_0 \dots x_{k-3}$
$k-1$	$x_0 \dots x_{k-2}$
Final	$x_1 \dots x_{k-1} = \beta_0 \dots \beta_{k-2}$

Table 2.6: Path traversed by loop.

Hence $x_i = \beta_{i-1}$ for $i = 1, \dots, k-1$.

We observe that x_0 can be either 0 or 1 and this gives the two sides of the loop.

Let a_i and b_i be the branch labels corresponding to the sides of the loop given by $x_0 = 0$ and $x_0 = 1$ respectfully.

The state transitions for each side of the loop are:

Side	Transitions
a	$0, \beta_0, \dots, \beta_{k-2}$
b	$1, \beta_0, \dots, \beta_{k-2}$

Table 2.7: Transitions along each side of loop.

Therefore

$$\begin{aligned} a_0 &= g_0 0 + g_1\alpha_{k-2} + g_2\alpha_{k-3} + \ldots + g_{k-1}\alpha_0 \\ b_0 &= g_0 1 + g_1\alpha_{k-2} + g_2\alpha_{k-3} + \ldots + g_{k-1}\alpha_0 \end{aligned} \tag{2.6.8}$$

Hence $b_0 - a_0 = g_0$.

For $i > 0$

$$\begin{aligned} a_i &= \sum_{j=0}^{i-1} g_j\beta_{i-j} + g_i 0 + \sum_{j=i+1}^{k-1} g_j\alpha_{k-1+i-j} \\ b_i &= \sum_{j=0}^{i-1} g_j\beta_{i-j} + g_i 1 + \sum_{j=i+1}^{k-1} g_j\alpha_{k-1+i-j} \end{aligned} \tag{2.6.9}$$

Hence $b_i - a_i = g_i$ for all $i = 0, \ldots, k-1$. □

The second invariant we consider allows us to determine the shortest path amongst a pair of loops of length K from an initial state to a pair of neighbouring final states. In particular, we make use of the following identity

$$\min(a, b) = \frac{1}{2}(a + b - |a - b|) \tag{2.6.10}$$

This allows us to calculate the difference between a pair of minimums as follows

$$\min(a, b) - \min(c, d) = \frac{1}{2}(a + b - c - d) - \frac{1}{2}(|a - b| - |c - d|) \tag{2.6.11}$$

Above we derived an invariant for absolute difference between branch labels, but we can also derive an invariant for the following branch metric relation $a_i + b_i - c_i - d_i$.

Theorem 2.6.2. Given a Viterbi decoder of constraint length k, with en-

coding given by polynomial $g_0 + g_1 D + \ldots + g_{k-1} D^{k-1}$ and two loops from state $\alpha_0 \ldots \alpha_{k-2}$ to states $\beta_0 \ldots \beta_{k-3} 0$ and $\beta_0 \ldots \beta_{k-3} 1$, then

$$a_i + b_i - c_i - d_i = \begin{cases} \pm 2 g_0 & \text{if } i = k-1 \\ 0 & \text{if } i = 0, \ldots, k-2 \end{cases} \tag{2.6.12}$$

where a_i, b_i, c_i, d_i are the branch labels corresponding to the sides of the loops as shown in figure 2.17.

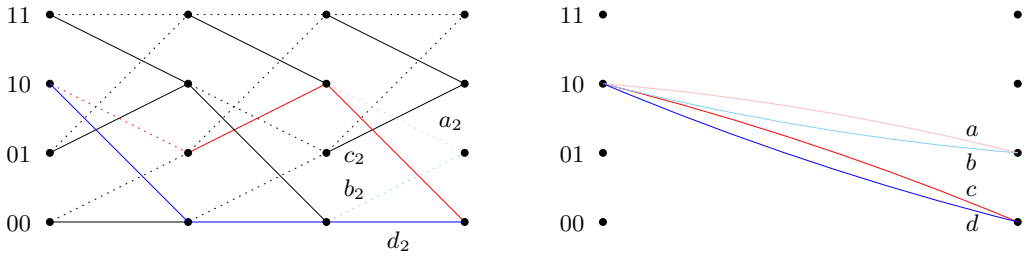


Figure 2.17: Two loops with same initial state and neighbouring final states.

Proof. A path through the trellis from the initial state to the final state must pass through $k-1$ other states between:

State Number	State	State
Initial	$\alpha_0 \ldots \alpha_{k-2}$	$\alpha_0 \ldots \alpha_{k-2}$
1	$\alpha_1 \ldots \alpha_{k-2} x_0$	$\alpha_1 \ldots \alpha_{k-2} y_0$
2	$\alpha_2 \ldots \alpha_{k-2} x_0 x_1$	$\alpha_2 \ldots \alpha_{k-2} y_0 y_1$
n	$\alpha_n \ldots \alpha_{k-2} x_0 \ldots x_{n-1}$	$\alpha_n \ldots \alpha_{k-2} y_0 \ldots y_{n-1}$
$k-2$	$\alpha_{k-2} x_0 \ldots x_{k-3}$	$\alpha_{k-2} y_0 \ldots y_{k-3}$
$k-1$	$x_0 \ldots x_{k-2}$	$y_0 \ldots y_{k-2}$
Final	$x_1 \ldots x_{k-1} = \beta_0 \ldots \beta_{k-3} 0$	$y_1 \ldots y_{k-1} = \beta_0 \ldots \beta_{k-3} 1$

Table 2.8: Path traversed by each loop.

Hence $x_{k-1} = 0$, $y_{k-1} = 1$, $x_i = \beta_{i-1}$ and $y_i = \beta_{i-1}$ for $i = 1, \ldots, k-2$.

We observe that x_0 can be either 0 or 1 and this gives the two sides of the first loop. Similarly that y_0 can be either 0 or 1 and this gives the two sides of the second loop.

Let a_i, b_i, c_i and d_i be the branch labels corresponding to the sides of the loops given by $y_0 = 1$, $y_0 = 0$, $x_0 = 1$ and $x_0 = 0$ respectfully.

The state transitions for each side of the loop are:

Side	Transitions
a	$1, \beta_0, \ldots, \beta_{k-3}, 1$
b	$0, \beta_0, \ldots, \beta_{k-3}, 1$
c	$1, \beta_0, \ldots, \beta_{k-3}, 0$
d	$0, \beta_0, \ldots, \beta_{k-3}, 0$

Table 2.9: Transitions along each side of each loop.

Therefore

$$\begin{aligned} a_{k-1} &= g_0 1 + g_1\beta_{k-3} + g_2\beta_{k-4} + \ldots + g_{k-2}\beta_0 + g_{k-1}1 \\ b_{k-1} &= g_0 1 + g_1\beta_{k-3} + g_2\beta_{k-4} + \ldots + g_{k-2}\beta_0 + g_{k-1}0 \\ c_{k-1} &= g_0 0 + g_1\beta_{k-3} + g_2\beta_{k-4} + \ldots + g_{k-2}\beta_0 + g_{k-1}1 \\ d_{k-1} &= g_0 0 + g_1\beta_{k-3} + g_2\beta_{k-4} + \ldots + g_{k-2}\beta_0 + g_{k-1}0 \end{aligned} \tag{2.6.13}$$

Hence $a_{k-1} + b_{k-1} - c_{k-1} - d_{k-1} = 2g_0$.

For $i = 0, \ldots, k-2$

$$\begin{aligned} a_i &= \sum_{j=0}^{i-1} g_j \beta_{i-j} + g_i 1 + \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \\ b_i &= \sum_{j=0}^{i-1} g_j \beta_{i-j} + g_i 0 + \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \\ c_i &= \sum_{j=0}^{i-1} g_j \beta_{i-j} + g_i 1 + \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \\ d_i &= \sum_{j=0}^{i-1} g_j \beta_{i-j} + g_i 0 + \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \end{aligned} \tag{2.6.14}$$

Hence $a_i + b_i - c_i - d_i = 0$ for all $i = 0, \ldots, k-2$. □

The third invariant we consider allows us to determine the shortest path amongst a pair of loops of length K from a pair of neighbouring initial states to a single final state.

Theorem 2.6.3. Given a Viterbi decoder of constraint length k, with encoding given by polynomial $g_0 + g_1 D + \ldots + g_{k-1} D^{k-1}$ and two loops from states $\alpha_0 \ldots \alpha_{k-3} 0$ and $\alpha_0 \ldots \alpha_{k-3} 1$ to state $\beta_0 \ldots \beta_{k-2}$, then

$$a_i + b_i - c_i - d_i = \begin{cases} \pm 2 g_{i-1} & \text{if } i = 0, \ldots, k-2 \\ 0 & \text{if } i = k-1 \end{cases} \tag{2.6.15}$$

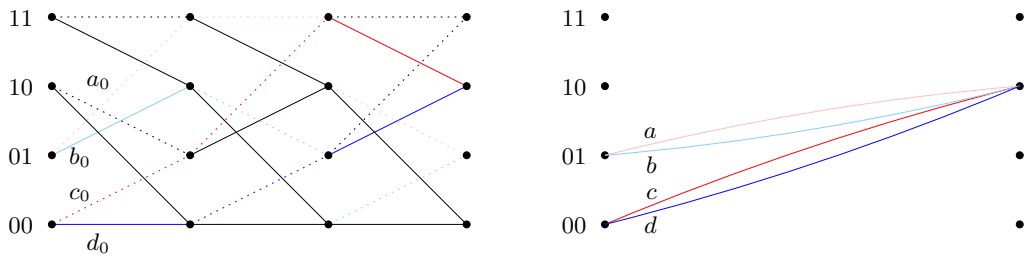


Figure 2.18: Two loops with same final state and neighbouring initial states.

Proof. A path through the trellis from the initial state to the final state must pass through $k-1$ other states between:

State Number	State	State
Initial	$\alpha_0 \ldots \alpha_{k-3}0$	$\alpha_0 \ldots \alpha_{k-3}1$
1	$\alpha_1 \ldots \alpha_{k-3}0x_0$	$\alpha_1 \ldots \alpha_{k-3}1y_0$
2	$\alpha_2 \ldots \alpha_{k-3}0x_0x_1$	$\alpha_2 \ldots \alpha_{k-3}1y_0y_1$
n	$\alpha_n \ldots \alpha_{k-3}0x_0 \ldots x_{n-1}$	$\alpha_n \ldots \alpha_{k-3}1y_0 \ldots y_{n-1}$
$k-3$	$\alpha_{k-3}0x_0 \ldots x_{k-3}$	$\alpha_{k-3}1y_0 \ldots y_{k-3}$
$k-2$	$0x_0 \ldots x_{k-3}$	$1y_0 \ldots y_{k-3}$
$k-1$	$x_0 \ldots x_{k-2}$	$y_0 \ldots y_{k-2}$
Final	$x_1 \ldots x_{k-1} = \beta_0 \ldots \beta_{k-2}$	$y_1 \ldots y_{k-1} = \beta_0 \ldots \beta_{k-2}$

Table 2.10: Path traversed by each loop.

Hence $x_i = \beta_{i-1}$ and $y_i = \beta_{i-1}$ for $i = 1, \ldots, k-1$.

We observe that x_0 can be either 0 or 1 and this gives the two sides of the first loop. Similarly that y_0 can be either 0 or 1 and this gives the two sides of the second loop.

Let a_i, b_i, c_i and d_i be the branch labels corresponding to the sides of the loops given by $y_0 = 1$, $y_0 = 0$, $x_0 = 1$ and $x_0 = 0$ respectfully.

The state transitions for each side of the loop are:

Side	Transitions
a	$1, \beta_0, \ldots, \beta_{k-2}$
b	$0, \beta_0, \ldots, \beta_{k-2}$
c	$1, \beta_0, \ldots, \beta_{k-2}$
d	$0, \beta_0, \ldots, \beta_{k-2}$

Table 2.11: Transitions along each side of each loop.

Therefore

$$\begin{aligned} a_{k-1} &= g_0\beta_{k-2} + g_1\beta_{k-3} + \ldots + g_{k-2}\beta_0 + g_{k-1}1 \\ b_{k-1} &= g_0\beta_{k-2} + g_1\beta_{k-3} + \ldots + g_{k-2}\beta_0 + g_{k-1}0 \\ c_{k-1} &= g_0\beta_{k-2} + g_1\beta_{k-3} + \ldots + g_{k-2}\beta_0 + g_{k-1}1 \\ d_{k-1} &= g_0\beta_{k-2} + g_1\beta_{k-3} + \ldots + g_{k-2}\beta_0 + g_{k-1}0 \end{aligned} \tag{2.6.16}$$

Hence $a_{k-1} + b_{k-1} - c_{k-1} - d_{k-1} = 0$.

For $i = 0, \ldots, k-2$

$$\begin{aligned} a_i &= \sum_{j=0}^{i-1} g_j\beta_{i-j} + g_i 1 + g_{i+1}1 + \sum_{j=i+2}^{k-1} g_j\alpha_{k-1+i-j} \\ b_i &= \sum_{j=0}^{i-1} g_j\beta_{i-j} + g_i 0 + g_{i+1}1 + \sum_{j=i+2}^{k-1} g_j\alpha_{k-1+i-j} \\ c_i &= \sum_{j=0}^{i-1} g_j\beta_{i-j} + g_i 1 + g_{i+1}0 + \sum_{j=i+2}^{k-1} g_j\alpha_{k-1+i-j} \\ d_i &= \sum_{j=0}^{i-1} g_j\beta_{i-j} + g_i 0 + g_{i+1}0 + \sum_{j=i+2}^{k-1} g_j\alpha_{k-1+i-j} \end{aligned} \tag{2.6.17}$$

Hence $a_i + b_i - c_i - d_i = 2g_{i+1}$ for all $i = 0, \ldots, k-2$. □

The final invariant we consider allows us to determine the shortest path

amongst a pair of loops of length K from an arbitrary pair of initial states to a single final state.

Theorem 2.6.4. Given a Viterbi decoder of constraint length k, with encoding given by polynomial $g_0 + g_1 D + \ldots + g_{k-1} D^{k-1}$ and two loops from states $\alpha_0 \ldots \alpha_{k-2}$ and $\beta_0 \ldots \beta_{k-2}$ to state $\gamma_0 \ldots \gamma_{k-2}$, then

$$a_i + b_i - c_i - d_i = \begin{cases} 2\sum_{j=i+1}^{k-1} g_j(\beta_{k-1+i-j} - \alpha_{k-1+i-j}) & \text{if } i = 0, \ldots, k-2 \\ 0 & \text{if } i = k-1 \end{cases} \tag{2.6.18}$$

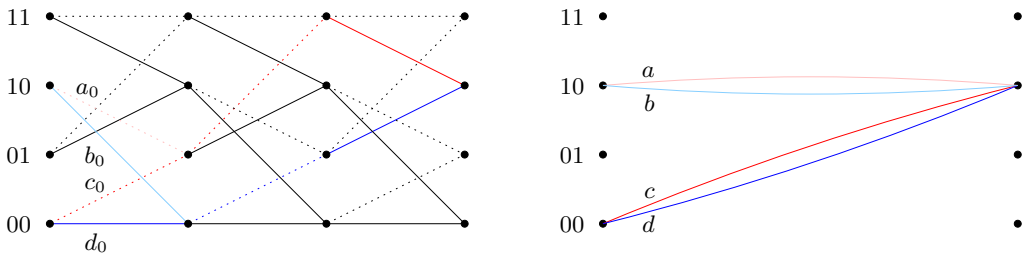


Figure 2.19: Two loops with same final state and any two initial states.

Proof. A path through the trellis from the initial state to the final state must pass through $k-1$ other states between:

State Number	State	State
Initial	$\alpha_0 \ldots \alpha_{k-2}$	$\beta_0 \ldots \beta_{k-2}$
1	$\alpha_1 \ldots \alpha_{k-2} x_0$	$\beta_1 \ldots \beta_{k-2} y_0$
2	$\alpha_2 \ldots \alpha_{k-2} x_0 x_1$	$\beta_2 \ldots \beta_{k-2} y_0 y_1$
n	$\alpha_n \ldots \alpha_{k-2} x_0 \ldots x_{n-1}$	$\beta_n \ldots \beta_{k-2} y_0 \ldots y_{n-1}$
$k-2$	$\alpha_{k-2} x_0 \ldots x_{k-3}$	$\beta_{k-2} y_0 \ldots y_{k-3}$
$k-1$	$x_0 \ldots x_{k-2}$	$y_0 \ldots y_{k-2}$
Final	$x_1 \ldots x_{k-1} = \gamma_0 \ldots \gamma_{k-2}$	$y_1 \ldots y_{k-1} = \gamma_0 \ldots \gamma_{k-2}$

Table 2.12: Path traversed by each loop.

Hence $x_i = \gamma_{i-1}$ and $y_i = \gamma_{i-1}$ for $i = 1, \ldots, k-1$.

We observe that x_0 can be either 0 or 1 and this gives the two sides of the first loop. Similarly that y_0 can be either 0 or 1 and this gives the two sides of the second loop.

Let a_i, b_i, c_i and d_i be the branch labels corresponding to the sides of the loops given by $y_0 = 1$, $y_0 = 0$, $x_0 = 1$ and $x_0 = 0$ respectfully.

The state transitions for each side of the loop are:

Side	Transitions
a	$1, \gamma_0, \ldots, \gamma_{k-2}$
b	$0, \gamma_0, \ldots, \gamma_{k-2}$
c	$1, \gamma_0, \ldots, \gamma_{k-2}$
d	$0, \gamma_0, \ldots, \gamma_{k-2}$

Table 2.13: Transitions along each side of each loop.

Therefore

$$\begin{aligned} a_{k-1} &= g_0\gamma_{k-2} + g_1\gamma_{k-3} + \ldots + g_{k-2}\gamma_0 + g_{k-1}1 \\ b_{k-1} &= g_0\gamma_{k-2} + g_1\gamma_{k-3} + \ldots + g_{k-2}\gamma_0 + g_{k-1}0 \\ c_{k-1} &= g_0\gamma_{k-2} + g_1\gamma_{k-3} + \ldots + g_{k-2}\gamma_0 + g_{k-1}1 \\ d_{k-1} &= g_0\gamma_{k-2} + g_1\gamma_{k-3} + \ldots + g_{k-2}\gamma_0 + g_{k-1}0 \end{aligned} \tag{2.6.19}$$

Hence $a_{k-1} + b_{k-1} - c_{k-1} - d_{k-1} = 0$.

For $i = 0, \ldots, k-2$

$$\begin{aligned} a_i &= \sum_{j=0}^{i-1} g_j \gamma_{i-j} + g_i 1 + \sum_{j=i+1}^{k-1} g_j \beta_{k-1+i-j} \\ b_i &= \sum_{j=0}^{i-1} g_j \gamma_{i-j} + g_i 0 + \sum_{j=i+1}^{k-1} g_j \beta_{k-1+i-j} \\ c_i &= \sum_{j=0}^{i-1} g_j \gamma_{i-j} + g_i 1 + \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \\ d_i &= \sum_{j=0}^{i-1} g_j \gamma_{i-j} + g_i 0 + \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \end{aligned} \tag{2.6.20}$$

Hence

$$\begin{aligned} a_i + b_i - c_i - d_i &= 2 \sum_{j=i+1}^{k-1} g_j \beta_{k-1+i-j} - 2 \sum_{j=i+1}^{k-1} g_j \alpha_{k-1+i-j} \\ &= 2 \sum_{j=i+1}^{k-1} g_j (\beta_{k-1+i-j} - \alpha_{k-1+i-j}) \end{aligned} \tag{2.6.21}$$

for all $i = 0, \ldots, k-2$. □

2.7 4-State White Noise Viterbi Detector Implementations

In this section we describe and compare various implementations of a programmable 4-state white noise Viterbi detector, where the ideal signal is given by

$$I(x_{i-2}x_{i-1}x_i) = g_0 x_i + g_1 x_{i-1} + g_2 x_{i-2} \tag{2.7.1}$$

where $x_k \in \{0, 1\}$.

We shall compare a standard 2T implementation against a 3T implementation with loop elimination and a 3T implementation with loop elimination and path invariants.

Note that as a result of loop elimination, all these implementations feature the same 2T ACS unit, therefore we only need consider the area of the branch metric unit.

2.7.1 Standard 2T Implementation

The branch metric unit for the standard 2T implementation is constructed by combining the outputs of 2 independent 1T BMU slices.

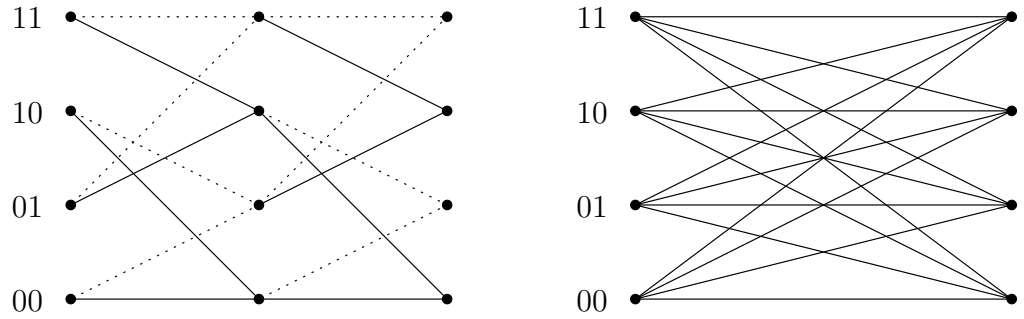


Figure 2.20: 4 state 2T trellis.

Each 1T BMU produces 8 branch metrics which must be summed together to produce the 16 branch metrics required for the higher radix implementation

according to the following rule

$$BM^{(T)}(x_{i-T-1}\dots x_i, r_{i-T+1}\dots r_i) = \sum_{k=0}^{T-1} BM^{(1)}(x_{i-k-2}x_{i-k-1}x_{i-k}, r_{i-k}) \tag{2.7.2}$$

for a T-T BMU.

The branch metric for a single 1T BMU slice is given by the following

$$BM^{(1)}(x_{i-2}x_{i-1}x_i, r_i) = I(x_{i-2}x_{i-1}x_i)^2 - 2r_iI(x_{i-2}x_{i-1}x_i). \tag{2.7.3}$$

Note that the $I(x_{i-2}x_{i-1}x_i)^2$ term is independent of the received signal, and can therefore be pre-programmed along with the coefficients g_k.

But the $r_iI(x_{i-2}x_{i-1}x_i)$ term does depend on the received signal. By calculating the following intermediate results using 3 multiplications and 4 additions

$$\begin{array}{lllll} M_0 = r_ig_0 & & M_1 = r_ig_1 & & M_2 = r_ig_2 \\ A_0 = M_0 + M_1 & A_1 = M_0 + M_2 & A_2 = M_1 + M_2 & A_3 = A_0 + M_2 \end{array} \tag{2.7.4}$$

we can calculate all 8 terms as follows

$$\begin{array}{llll} r_iI(000) = 0 & r_iI(001) = M_0 & r_iI(010) = M_1 & r_iI(011) = A_0 \\ r_iI(100) = M_2 & r_iI(101) = A_1 & r_iI(110) = A_2 & r_iI(111) = A_3 \end{array} \tag{2.7.5}$$

To combine the received signal dependent and independent terms requires a further 7 additions[1].

[1]One of the additions is degenerate since at least one of the operands is zero, therefore only 7 of the 8 additions are non-trivial.

Therefore the total number of operation required to compute all branch metrics in a 1T BMU slice is 3 multiplications and 11 additions.

The 2T BMU therefore consists of 6 multiplications and 22 additions for the 1T BMU slices, plus 13 additions[2] to combine the 1T branch metrics together to form the 16 2T branch metrics.

The following table summarises the complexity of the standard 2T implementation.

BMU Implementation	Adders	Multipliers	Outputs/Cycle
2T	35	6	2

Table 2.14: Complexity of 2T implementation.

2.7.2 3T Implementation with Loop Elimination

The branch metric unit for the 3T implementation with loop elimination is constructed by combining the outputs of 3 independent 1T BMU slices, which produces 32 branch metrics arranged in 16 loops. The loops are then eliminated, and the remaining 16 branch metrics are fed to the 2T ACS unit.

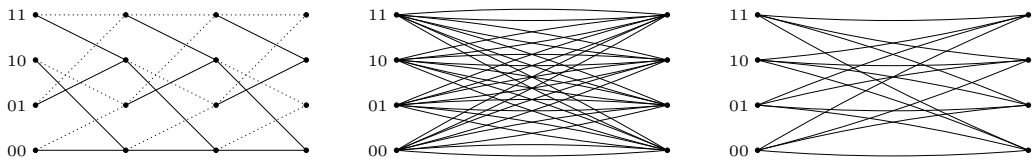


Figure 2.21: 4 state 3T trellis.

To combine the branch metrics from the three independent 1T BMU slices,

[2]Three of the additions are degenerate since at least one of the operands is zero, therefore only 13 of the 16 additions are non-trivial.

we first combine a pair of 1T BMU slices as we did for the standard 2T implementation.

The 16 branch metrics from the 2T BMU slice must then be combined with the 8 branch metrics from the remaining 1T BMU slice to produce the 32 branch metrics. This requires 27 additions[3] which means a total of 3 multiplications and 38 additions are required in addition to the standard 2T BMU to generate the 32 branch metrics.

Finally we need to eliminate the loops by comparing the branch metrics which form the sides of each loop. This requires a further 15 minimisations[4], which we shall count as 15 additions since the complexity of an addition and minimisation is comparable.

The following table summarises the complexity of the 2T standard implementation and the 3T implementation with loop elimination.

BMU Implementation	Adders	Multipliers	Outputs/Cycle
2T	35	6	2
3T + Loops	88	9	3

Table 2.15: Complexity of 2T standard implementation and 3T implementation with loop elimination.

[3]Five of the additions are degenerate since at least one of the operands is zero, therefore only 27 of the 32 additions are non-trivial.

[4]One of the minimisations is degenerate since at least one of the operands is zero, therefore only 15 of the 16 minimisations are non-trivial.

2.7.3 3T Implementation with Loop Elimination and Path Invariants

With the use of the invariants we proved earlier in this chapter, we can reduce the complexity of the 3T BMU with loop elimination.

In the 3T BMU with loop elimination, we find the shortest of the two paths from each initial state $x_{i-4}x_{i-3}$ to each final state $x_{i-1}x_i$.

$$BE(x_{i-4}x_{i-3}, x_{i-1}x_i) = \min\left(BM^{(3)}(x_{i-4}x_{i-3}0x_{i-1}x_i), BM^{(3)}(x_{i-4}x_{i-3}1x_{i-1}x_i)\right) \tag{2.7.6}$$

To take advantage of invariants, we rewrite the above expression in terms of differences between the sides of loops and relative to the zero initial state.

$$\begin{aligned} BE(x_{i-4}x_{i-3}, x_{i-1}x_i) = BM^{(3)}(000x_{i-1}x_i) + D(x_{i-4}x_{i-3}, x_{i-1}x_i) \\ + \max(0, \Delta(x_{i-4}x_{i-3}, x_{i-1}x_i)) \end{aligned} \tag{2.7.7}$$

where

$$\begin{aligned} D(x_{i-4}x_{i-3}, x_{i-1}x_i) &= BM^{(3)}(x_{i-4}x_{i-3}0x_{i-1}x_i) - BM^{(3)}(000x_{i-1}x_i) \\ \Delta(x_{i-4}x_{i-3}, x_{i-1}x_i) &= BM^{(3)}(x_{i-4}x_{i-3}1x_{i-1}x_i) - BM^{(3)}(x_{i-4}x_{i-3}0x_{i-1}x_i) \end{aligned} \tag{2.7.8}$$

We can then use (2.6.7) to compute

$$\Delta(x_{i-4}x_{i-3}, x_{i-1}x_i) = G(x_{i-4}x_{i-3}, x_{i-1}x_i) - 2\sum_{k=0}^{2} g_{2-k}r_{i-k} \tag{2.7.9}$$

where

$$G(x_{i-4}x_{i-3}, x_{i-1}x_i) = \sum_{k=0}^{2} \left(I(x_{i-k-2}x_{i-k-1}x_{i-k})^2 \mid_{x_{i-2}=1} - I(x_{i-k-2}x_{i-k-1}x_{i-k})^2 \mid_{x_{i-2}=0} \right) \tag{2.7.10}$$

If we explicitly expand the term which is independent of the received signal we find that

$$G(x_{i-4}x_{i-3}, x_{i-1}x_i) = g_0^2 + g_1^2 + g_2^2 + 2(g_0g_1 + g_1g_2)(x_{i-3} + x_{i-1}) + 2g_0g_2(x_{i-4} + x_i) \tag{2.7.11}$$

As a result there are only 9 different values this term can take since it depends only on $x_{i-3} + x_{i-1}, x_{i-4} + x_i \in \{0, 1, 2\}$. These 9 values are pre-programmed along with the coefficients g_k.

Note that the term which depends on the received signal is independent of both the initial and final states, and can be implemented in 3 multiplications and 2 additions

$$\sum_{k=0}^{2} g_{2-k} r_{i-k} = g_2 r_i + g_1 r_{i-1} + g_0 r_{i-2} \tag{2.7.12}$$

A further 9 additions are required to add the term dependent on the received signal to each of the 9 distinct terms independent of the received signal, giving a total of 3 multiplications and 11 additions to compute all the $\Delta(x_{i-4}x_{i-3}, x_{i-1}x_i)$ terms.

Note that in twos complement arithmetic, the most significant bit repre-

sents the sign. Therefore by using a bitwise AND of each Δ with the inverse of its own sign bit, we set negative values to zero and thus obtain $\max(0, \Delta(x_{i-4}x_{i-3}, x_{i-1}x_i))$.

Next we compute

$$\begin{aligned} BM^{(3)}(000x_{i-1}x_i) &= BM^{(1)}(000) + BM^{(1)}(00x_{i-1}) + BM^{(1)}(0x_{i-1}x_i) \\ &= \left((g_0^2 + g_1^2)x_{i-1}^2 + 2g_1g_0x_{i-1}x_i + g_0^2x_i^2\right) \\ &\quad - (2(r_{i-1}g_0 + r_ig_1)x_{i-1} + 2r_ig_0x_i) \end{aligned} \tag{2.7.13}$$

Since $x_k \in \{0, 1\}$, we can compute all 4 terms which depend on the received signal using 3 multiplications and 2 additions, whilst the 4 distinct values for the term which is independent of the received signal can be pre-programmed along with the coefficients g_k.

The terms dependent on the received signal can then be added to the terms independent of the received signal using a further 3 additions[5], resulting in a total of 3 multiplications and 5 additions to compute all $BM^{(3)}(000x_{i-1}x_i)$ terms.

Finally, we compute

$$\begin{aligned} D(x_{i-4}x_{i-3}, x_{i-1}x_i) &= BM^{(3)}(x_{i-4}x_{i-3}0x_{i-1}x_i) - BM^{(3)}(000x_{i-1}x_i) \\ &= \left((g_1^2 + g_2^2)x_{i-3}^2 + g_2^2x_{i-4}^2 + 2g_1g_2x_{i-3}x_{i-4} + 2g_0g_2x_{i-1}x_{i-3}\right) \\ &\quad - (2(r_{i-2}g_1 + 2r_{i-1}g_2)x_{i-3} + 2r_{i-2}g_2x_{i-4}) \end{aligned} \tag{2.7.14}$$

[5]One of the additions is degenerate since at least one of the operands is zero, therefore only 3 of the 4 additions are non-trivial.

As before, since $x_k \in \{0, 1\}$, we can compute all 4 terms which depend on the received signal using 3 multiplications and 2 additions, whilst the 6 distinct values for the term which is independent of the received signal can be pre-programmed along with the coefficients g_k.

The terms dependent on the received signal can then be added to the terms independent of the received signal using a further 5 additions[6], resulting in a total of 3 multiplications and 7 additions to compute all $D(x_{i-4}x_{i-3}, x_{i-1}x_i)$ terms.

We are now ready to sum the $BM^{(3)}(000x_{i-1}x_i)$ with the $D(x_{i-4}x_{i-3}, x_{i-1}x_i)$ terms. This requires only 9 additions since 5 of the possible 16 additions is degenerate since at least one of the operands is zero.

Finally we must add the $\Delta(x_{i-4}x_{i-3}, x_{i-1}x_i)$ terms, which requires a further 15 non-trivial additions.

The following table summarises the complexity of each implementation we have considered.

BMU Implementation	Adders	Multipliers	Outputs/Cycle
2T	35	6	2
3T + Loops	88	9	3
3T + Loops + Invariants	47	9	3

Table 2.16: Complexity of various 4-state implementations.

In summary, we have demonstrated an architecture for a 3T 4-state programmable Viterbi detector which is 50% faster than the standard 2T im-

[6]One of the additions is degenerate since at least one of the operands is zero, therefore only 3 of the 4 additions are non-trivial.

plementation, yet has exactly the same ACS unit and a BMU which is less than 50% larger.

This compares favourably to the standard 3T implementation in which the ACS unit alone is at least twice the size of the standard 2T implementation, and only offers a modest improvement in performance due to the extra comparison on the ACS unit critical path.

Chapter 3

High Throughput Viterbi Decoders

The Viterbi decoders used in communications systems are essentially the same as Viterbi detectors used in read channel, but the convolutional codes are not determined by the channel as in read channel, but they are chosen to meet the performance requirements, and there may be multiple encoder outputs for each input. Modulo 2 arithmetic is also used.

For example, consider the $K = 7$ rate $\frac{1}{2}$ convolutional code octal (171,133) described below, which is used various industry standards including Q1900, DVB, IEEE802.11a/b/g/n [22], IEEE802.16a [23], HiperAccess, HiperMan and INTELSAT IESS-308/309.

This code has two output bits $y_k^{(0)}, y_k^{(1)}$ for each input bit x_k.

$$\begin{aligned} y_k^{(0)} &= x_k \oplus x_{k-1} \oplus x_{k-2} \oplus x_{k-3} \oplus x_{k-6} \\ y_k^{(1)} &= x_k \oplus x_{k-2} \oplus x_{k-3} \oplus x_{k-5} \oplus x_{k-6} \end{aligned} \tag{3.0.1}$$

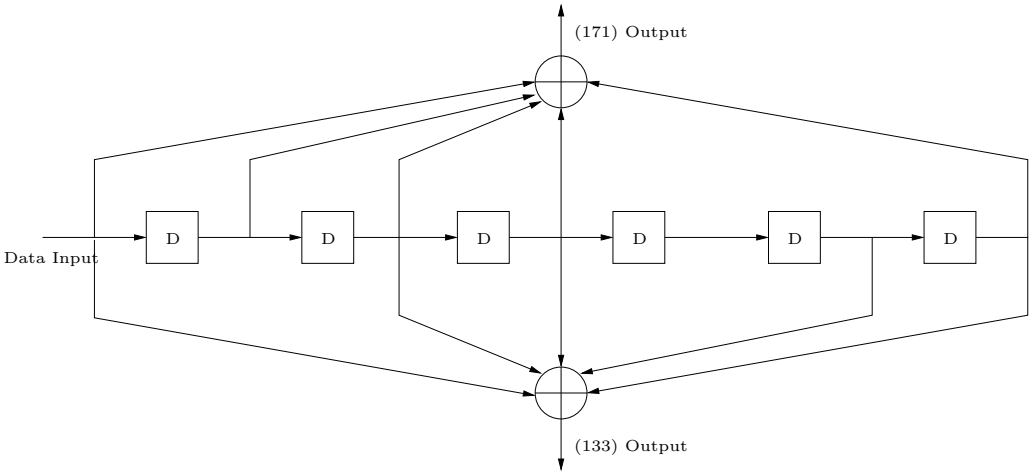


Figure 3.1: $K = 7$ rate $\frac{1}{2}$ convolutional code octal (171,133).

In this chapter we introduce invariants to reduce the complexity of Viterbi decoders, give a tight bound on the complexity and then apply the results to the industry standard code described above.

3.1 Loop Invariants in Communications

Suppose we encode the original data with the convolutional code

$$y_k = g_0 x_k \oplus g_1 x_{k-1} \oplus \ldots \oplus g_{K-1} x_{k-K+1} \tag{3.1.1}$$

The Euclidean branch metric is given by

$$b_{\text{Euclidean}} = (r - y_{\text{NRZ}})^2 \tag{3.1.2}$$

where $y_{\text{NRZ}} \in \{\pm 1\}$. Let $y = \frac{1}{2}(y_{\text{NRZ}} + 1)$ such that $y \in \{0, 1\}$, then

$$\begin{aligned} b_{\text{Euclidean}} &= (r - y_{\text{NRZ}})^2 \\ &= r^2 + y_{\text{NRZ}}^2 - 2ry_{\text{NRZ}} \\ &= r^2 + 1 - 2r(2y - 1) \\ &= (r + 1)^2 - 4ry \end{aligned} \tag{3.1.3}$$

Since the term $(r + 1)^2$ is common to all branches, it can be neglected as it has no effect when comparing branches in order to find the minimum. Similarly the multiplicative factor 4 may also be neglected. Therefore by removing common terms and scalar constants, we can choose the following branch metric

$$b' = -ry \tag{3.1.4}$$

The branch metric can be simplified further by changing its sign. However negation of the branch metrics requires the path metric to be calculated as the maximum sum of branch metrics, rather than the minimum. Therefore we choose the following branch metric

$$b = ry = \begin{cases} 0 & \text{if } y = 0 \\ r & \text{if } y = 1 \end{cases} \tag{3.1.5}$$

which selects between the received signal and zero depending on the ideal output, and note that we must maximise the path metric.

Consider a loop of length K from initial state $i_1 i_2 \dots i_{K-1}$ to final state $f_1 f_2 \dots f_{K-1}$. Let $B^{(1)}$ and $B^{(0)}$ be the branch metrics corresponding to the top and bottom sides of the loop, where the top side of the loop corresponds to the path through state $i_2 \dots i_{K-1}1$ and the bottom side of the loop corresponds to the path through state $i_2 \dots i_{K-1}0$.

To eliminate one side of the loop, we need to compute $\max(B^{(1)}, B^{(0)})$ which we can do using the following identity

$$\max(B^{(1)}, B^{(0)}) = \frac{1}{2}(B^{(1)} + B^{(0)}) + \frac{1}{2}\left|B^{(1)} - B^{(0)}\right| \tag{3.1.6}$$

Therefore we need to consider how to calculate branch metric sums and differences on the loop.

$$\begin{aligned} B^{(1)} &= \sum_{k=0}^{K-1} b_k^{(1)} = \sum_{k=0}^{K-1} r_k y_k^{(1)} \\ B^{(0)} &= \sum_{k=0}^{K-1} b_k^{(0)} = \sum_{k=0}^{K-1} r_k y_k^{(0)} \end{aligned} \tag{3.1.7}$$

Consider the ideal output at k-th step ($0 \le k \le K-1$). The state to the right of the k-th time slice for the upper and lower paths is

$$\begin{aligned} s_k^{(1)} &= \{i_{k+2} \dots i_{K-1} 1 f_1 \dots f_k\} \\ s_k^{(0)} &= \{i_{k+2} \dots i_{K-1} 0 f_1 \dots f_k\} \end{aligned} \tag{3.1.8}$$

and the ideal outputs are

$$\begin{aligned} y_k^{(1)} &= (g_0 f_k \oplus \ldots \oplus g_{k-1} f_1) \oplus 1 \cdot g_k \oplus (g_{k+1} i_{K-1} \oplus \ldots \oplus g_{K-1} i_{k+2}) \\ y_k^{(0)} &= (g_0 f_k \oplus \ldots \oplus g_{k-1} f_1) \oplus 0 \cdot g_k \oplus (g_{k+1} i_{K-1} \oplus \ldots \oplus g_{K-1} i_{k+2}) \end{aligned} \tag{3.1.9}$$

Therefore

$$y_k^{(1)} \oplus y_k^{(0)} = g_k \tag{3.1.10}$$

Hence

$$y_k^{(1)} = \begin{cases} y_k^{(0)} & \text{if } g_k = 0 \\ \overline{y_k^{(0)}} & \text{if } g_k = 1 \end{cases} \tag{3.1.11}$$

Define the sums and differences of branch metrics along a loop as

$$\begin{aligned} b_k^{(+)} &= b_k^{(1)} + b_k^{(0)} = (y_k^{(1)} + y_k^{(0)}) r_k \\ b_k^{(-)} &= b_k^{(1)} - b_k^{(0)} = (y_k^{(1)} - y_k^{(0)}) r_k \end{aligned} \tag{3.1.12}$$

Then

$$\begin{aligned} b_k^{(+)} &= \begin{cases} 2y_k^{(0)} r_k & \text{if } g_k = 0 \\ r_k & \text{if } g_k = 1 \end{cases} \\ b_k^{(-)} &= \begin{cases} 0 & \text{if } g_k = 0 \\ (-1)^{y_k^{(0)}} r_k & \text{if } g_k = 1 \end{cases} \end{aligned} \tag{3.1.13}$$

Note that the actual branch metric sum and difference between the two sides

of a loop is given by

$$\begin{aligned} B^{(1)} + B^{(0)} &= \sum_{k=0}^{K-1} b_k^{(+)} \\ B^{(1)} - B^{(0)} &= \sum_{k=0}^{K-1} b_k^{(-)} \end{aligned} \tag{3.1.14}$$

Therefore the total number of distinct branch metric sums ($B^{(1)} + B^{(0)}$) and differences ($B^{(1)} - B^{(0)}$), which we will denote $N^{(+)}$ and $N^{(-)}$ respectfully, is equal to the number of distinct vectors $\underline{b}^{(+)}$ and $\underline{b}^{(-)}$, where

$$\begin{aligned} \underline{b}^{(+)} &= \left\{ b_0^{(+)}, \ldots, b_{K-1}^{(+)} \right\} \\ \underline{b}^{(-)} &= \left\{ b_0^{(-)}, \ldots, b_{K-1}^{(-)} \right\} \end{aligned} \tag{3.1.15}$$

Since when $g_k = 1$ the corresponding branch metric sum is exactly the received signal r_k (which fixed within a time slice), distinctions between vectors $\underline{b}^{(+)}$ only occur when $g_k = 0$, and in this case $b_k^{(+)}$ still only takes one of two possible values 0 or $2r_k$ depending on the binary value $y_k^{(0)}$.

Similarly since when $g_k = 0$ the corresponding branch metric difference is zero, distinctions between vectors $\underline{b}^{(-)}$ only occur when $g_k = 1$, and in this case $b_k^{(-)}$ still only takes one of two possible values $\pm r_k$ depending on the binary value $y_k^{(0)}$.

Therefore from the explanation above and (3.1.13) it is clear that

$$\begin{aligned} N^{(+)} &\leq 2^{K - \mathrm{Ham}(\underline{g})} \\ N^{(-)} &\leq 2^{\mathrm{Ham}(\underline{g})} \end{aligned} \tag{3.1.16}$$

Example 3.1.1. Suppose $g_0 = 1$ and $g_i = 0$ for $i > 0$. Then

$$\begin{aligned} \underline{y}^{(1)} &= \{1, f_1, f_2, \ldots, f_{K-1}\} \\ \underline{y}^{(0)} &= \{0, f_1, f_2, \ldots, f_{K-1}\} \end{aligned} \tag{3.1.17}$$

$$\begin{aligned} \underline{b}^{(+)} &= \{1, f_1, f_2, \ldots, f_{K-1}\} \\ \underline{b}^{(-)} &= \{1, 0, 0, \ldots, 0\} \end{aligned} \tag{3.1.18}$$

Therefore there are 2^{K-1} distinct $\underline{b}^{(+)}$ and a single $\underline{b}^{(-)}$.

$$\begin{aligned} N^{(+)} &= 2^{K-1} \\ N^{(-)} &= 1 < 2 \end{aligned} \tag{3.1.19}$$

where 2 is the upper bound for $N^{(-)}$ given by (3.1.16).

Example 3.1.2. Suppose $g_i = 1$ for all i. Then

$$\begin{aligned} \underline{y}^{(1)} = \{&1 \oplus i_{K-1} \oplus i_{K-2} \oplus \ldots \oplus i_1, f_1 \oplus 1 \oplus i_{K-1} \oplus i_{K-2} \oplus \ldots \oplus i_2, \\ &f_2 \oplus f_1 \oplus 1 \oplus i_{K-1} \oplus i_{K-2} \oplus \ldots \oplus i_3, \ldots, f_{K-1} \oplus f_{K-2} \oplus \ldots \oplus f_1 \oplus 1\} \\ \underline{y}^{(0)} = \{&i_{K-1} \oplus i_{K-2} \oplus \ldots \oplus i_1, f_1 \oplus i_{K-1} \oplus i_{K-2} \oplus \ldots \oplus i_2, \\ &f_2 \oplus f_1 \oplus i_{K-1} \oplus i_{K-2} \oplus \ldots \oplus i_3, \ldots, f_{K-1} \oplus f_{K-2} \oplus \ldots \oplus f_1\} \end{aligned} \tag{3.1.20}$$

$$\begin{aligned} \underline{b}^{(+)} &= \{1, 1, \ldots, 1\} \\ \underline{b}^{(-)} &= \left\{(-1)^{y_0^{(0)}}, (-1)^{y_1^{(0)}}, \ldots, (-1)^{y_{K-1}^{(0)}}\right\} \end{aligned} \tag{3.1.21}$$

For example when $K = 3$,

$$\begin{aligned} \underline{y}^{(1)} &= \{1 \oplus i_2 \oplus i_1, 1 \oplus f_1 \oplus i_2, 1 \oplus f_2 \oplus f_1\} \\ \underline{y}^{(0)} &= \{i_2 \oplus i_1, f_1 \oplus i_2, f_2 \oplus f_1\} \end{aligned} \tag{3.1.22}$$

$$\begin{aligned} \underline{b}^{(+)} &= \{1, 1, 1\} \\ \underline{b}^{(-)} &= \left\{(-1)^{i_2 \oplus i_1}, (-1)^{f_1 \oplus i_2}, (-1)^{f_2 \oplus f_1}\right\} \end{aligned} \tag{3.1.23}$$

Therefore there are 2^K distinct $\underline{b}^{(-)}$ and a single $\underline{b}^{(+)}$.

$$\begin{aligned} N^{(+)} &= 1 \\ N^{(-)} &= 2^K \end{aligned} \tag{3.1.24}$$

which are exactly the upper bounds given by (3.1.16).

Now consider a rate $\frac{1}{N}$ code with generator matrix

$$G = \begin{pmatrix} \underline{g}_0 \\ \underline{g}_1 \\ \vdots \\ \underline{g}_{N-1} \end{pmatrix} = \begin{pmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,K-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,K-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N-1,0} & g_{N-1,1} & \cdots & g_{N-1,K-1} \end{pmatrix} \tag{3.1.25}$$

which is used to generate N separate convolutional encodings of the original data

$$y_{n,k} = g_{n,0}x_k \oplus g_{n,1}x_{k-1} \oplus \ldots \oplus g_{n,K-1}x_{k-K+1} \tag{3.1.26}$$

for $n = 0, \ldots, N-1$, then the branch metric, b_k, is given by

$$b_k = \sum_{n=0}^{N-1} b_{n,k} = \sum_{n=0}^{N-1} r_{n,k} y_{n,k} \tag{3.1.27}$$

Hence the branch metrics along the top and bottom of loops can be expressed as

$$\begin{aligned} B^{(1)} &= \sum_{k=0}^{K-1} b_k^{(1)} = \sum_{k=0}^{K-1} \sum_{n=0}^{N-1} r_{n,k} y_{n,k}^{(1)} \\ B^{(0)} &= \sum_{k=0}^{K-1} b_k^{(0)} = \sum_{k=0}^{K-1} \sum_{n=0}^{N-1} r_{n,k} y_{n,k}^{(0)} \end{aligned} \tag{3.1.28}$$

Therefore we can calculate the branch metric sums and differences corresponding to each vector of G independently, then sum the results.

Let $N_n^{(+)}$ be the number of distinct branch metric sums corresponding to vector $\underline{g}_n$ and let $N_n^{(-)}$ be the number of distinct branch metric differences corresponding to vector $\underline{g}_n$. Then the total number of distinct branch metric sums and differences for the matrix G is given by

$$\begin{aligned} N^{(+)} &= \prod_{n=0}^{N-1} N_n^{(+)} \\ N^{(-)} &= \prod_{n=0}^{N-1} N_n^{(-)} \end{aligned} \tag{3.1.29}$$

Using (3.1.16), we can bound $N^{(+)}$ and $N^{(-)}$ as

$$\begin{aligned} N^{(+)} &\leq 2^{NK - \mathrm{Ham}(G)} \\ N^{(-)} &\leq 2^{\mathrm{Ham}(G)} \end{aligned} \tag{3.1.30}$$

To compute the maximum of the branch metrics along a loop, we need to sum

the branch metric sums and differences. There are a maximum of $N^{(+)} \cdot N^{(-)}$ distinct maximum branch metrics along a loop, however if the total number of loops, 2^{2K-2}, is less than this, then we only need compute 2^{2K-2} combinations of branch metric sums and differences. Therefore the number of additions required to compute the maximum of the branch metrics along a loop from the branch metric sums and differences is given by

$$C_M = \min(2^{2K-2}, N^{(+)} \cdot N^{(-)}) \tag{3.1.31}$$

Note that for rate-1 codes, $N^{(+)} \cdot N^{(-)}$ is bounded by

$$N^{(+)} \cdot N^{(-)} \leq 2^{K-\mathrm{Ham}(G)} 2^{\mathrm{Ham}(G)} = 2^K \tag{3.1.32}$$

Therefore

$$C_M \leq 2^K \tag{3.1.33}$$

When $N > 1$ for rate-$\frac{1}{N}$ codes, $N^{(+)} \cdot N^{(-)}$ is bounded by

$$N^{(+)} \cdot N^{(-)} \leq 2^{NK-\mathrm{Ham}(G)} 2^{\mathrm{Ham}(G)} = 2^{NK} \tag{3.1.34}$$

Therefore

$$C_M \leq \min(2^{2K-2}, 2^{NK}) = 2^{2K-2} \tag{3.1.35}$$

for $K > 1$.

Example 3.1.3. Consider a rate $\frac{1}{2}$ code with generator matrix, G, made out

of vectors $\underline{g}_0$ and $\underline{g}_1$ from the previous two examples.

$$G = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 \end{pmatrix} \tag{3.1.36}$$

We previously calculated that there are 2^{K-1} distinct $\underline{b}_0^{(+)}$, a single $\underline{b}_0^{(-)}$, 2^K distinct $\underline{b}_1^{(-)}$ and a single $\underline{b}_1^{(+)}$.

$$\begin{aligned} N_0^{(+)} &= 2^{K-1} \\ N_0^{(-)} &= 1 \\ N_1^{(+)} &= 1 \\ N_1^{(-)} &= 2^K \end{aligned} \tag{3.1.37}$$

Therefore there are 2^{K-1} distinct $\underline{b}^{(+)}$ and 2^K distinct $\underline{b}^{(-)}$.

$$\begin{aligned} N^{(+)} &= 2^{K-1} \\ N^{(-)} &= 2^K < 2^{K+1} \end{aligned} \tag{3.1.38}$$

where 2^{K+1} is the upper bound for $N^{(-)}$ given by (3.1.30).

Example 3.1.4. Suppose $K = 3$ and the generator matrix is given by

$$G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \tag{3.1.39}$$

Then

$$\begin{aligned}
\underline{y}_0^{(1)} &= \{1 \oplus i_2 \oplus i_1, f_1 \oplus 1 \oplus i_2, f_2 \oplus f_1 \oplus 1\} \\
\underline{y}_0^{(0)} &= \{i_2 \oplus i_1, f_1 \oplus i_2, f_2 \oplus f_1\} \\
\underline{y}_1^{(1)} &= \{1 \oplus i_2, f_1 \oplus 1, f_2 \oplus f_1\} \\
\underline{y}_1^{(0)} &= \{i_2, f_1, f_2 \oplus f_1\}
\end{aligned} \tag{3.1.40}$$

$$\begin{aligned}
\underline{b}_0^{(+)} &= \{1, 1, 1\} \\
\underline{b}_0^{(-)} &= \left\{(-1)^{i_2 \oplus i_1}, (-1)^{f_1 \oplus i_2}, (-1)^{f_1 \oplus f_2}\right\} \\
\underline{b}_1^{(+)} &= \{1, 1, 2(f_2 \oplus f_1)\} \\
\underline{b}_1^{(-)} &= \left\{(-1)^{i_2}, (-1)^{f_1}, 0\right\}
\end{aligned} \tag{3.1.41}$$

Note that fixing $\underline{b}_1^{(-)}$, i.e. fixing i_2 and f_1, we are left with only 4 possible choices from the original 8 for $\underline{b}_0^{(-)}$. Therefore there are 2 distinct $\underline{b}^{(+)}$ and 16 distinct $\underline{b}^{(-)}$.

$$\begin{aligned}
N^{(+)} &= 2 \\
N^{(-)} &= 16 < 32
\end{aligned} \tag{3.1.42}$$

where 32 is the upper bound for $N^{(-)}$ given by (3.1.30).

Example 3.1.5. Suppose $K = 3$ and the generator matrix is given by

$$G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{3.1.43}$$

$\underline{b}_0^{(+)}$ and $\underline{b}_0^{(-)}$ are the same as the previous example. Also

$$\begin{aligned}
\underline{y}_1^{(1)} &= \{1, f_1, f_2\} \\
\underline{y}_1^{(0)} &= \{0, f_1, f_2\}
\end{aligned} \tag{3.1.44}$$

$$\begin{aligned} \underline{b}_1^{(+)} &= \{1, 2f_1, 2f_2\} \\ \underline{b}_1^{(-)} &= \{1, 0, 0\} \end{aligned} \tag{3.1.45}$$

Therefore there are 4 distinct $\underline{b}^{(+)}$ and also 4 distinct $\underline{b}^{(-)}$.

$$\begin{aligned} N^{(+)} &= 4 \\ N^{(-)} &= 4 < 16 \end{aligned} \tag{3.1.46}$$

where 16 is the upper bound for $N^{(-)}$ given by (3.1.30).

In (2.5.9) we gave a bound for the number of operations required per K-T time step to calculate the branch metrics and eliminate the loops. We now consider the operation count when using loop invariants. From (3.1.6), in order to eliminate the loops, we must calculate the following for each loop

$$\max(B^{(1)}, B^{(0)}) = \frac{1}{2}(B^{(1)} + B^{(0)}) + \frac{1}{2}\left|B^{(1)} - B^{(0)}\right| \tag{3.1.47}$$

In order to calculate the above, we compute all distinct branch metric sums and absolute differences, then sum combinations of these to calculate the maximum branch metric along a loop. We shall refer to the operation count for each of the above as C_S, C_D and C_M respectfully. In (3.1.33) and (3.1.35), we bounded C_M as

$$C_M \leq \begin{cases} 2^K & \text{for } N = 1 \\ 2^{2K-2} & \text{for } N > 1 \end{cases} \tag{3.1.48}$$

Next we consider how many additions, C_S, are required to calculate all the

distinct branch metric sums, $\underline{b}^{(+)}$. Recall from (3.1.13) that

$$b_{n,k}^{(+)} = \begin{cases} 2y_{n,k}^{(0)} r_{n,k} & \text{if } g_{n,k} = 0 \\ r_{n,k} & \text{if } g_{n,k} = 1 \end{cases} \tag{3.1.49}$$

Each distinct $\underline{b}^{(+)}$ contains the following invariant sum

$$\sum_{n,k|g_{n,k}=1} b_{n,k}^{(+)} = \sum_{n,k|g_{n,k}=1} r_{n,k} \tag{3.1.50}$$

which requires $C_{SI} = \text{Ham}(G) - 1$ additions to compute.

Each distinct $\underline{b}^{(+)}$ also contains the following varying sum

$$\sum_{n,k|g_{n,k}=0} b_{n,k}^{(+)} = \sum_{n,k|g_{n,k}=0} \begin{cases} 0 & \text{if } y_{n,k}^{(0)} = 0 \\ 2r_{n,k} & \text{if } y_{n,k}^{(0)} = 1 \end{cases} \tag{3.1.51}$$

in which there are $NK - \text{Ham}(G)$ terms in the summation. For simplicity, suppose that $T = NK - \text{Ham}(G)$ is a power of 2, then we can form all possible summations using a binary tree sharing all intermediate nodes in a similar manner to (2.5.6). This requires C_{SV} additions

$$C_{SV} = T \sum_{i=1}^{\log_2 T} \frac{(2^{2^{i-1}} - 1)^2}{2^i} \tag{3.1.52}$$

Reapplying the upper bound determined in (2.5.8), we can bound C_{SV} as

follows

$$\begin{aligned} C_{SV} &\leq T \sum_{i=1}^{\log_2 T} \frac{2^{2^i}}{2^i} \\ &\leq 2^T + \left(\log_2 \frac{T}{2}\right) 2^{\frac{T}{2}+1} \end{aligned} \tag{3.1.53}$$

To combine the invariant and varying sums requires a further $2^{NK-\mathrm{Ham}(G)}$ additions (one for each varying sum). Therefore the total addition count for calculating all the possible distinct branch metric sums is

$$\begin{aligned} C_S &\leq (NK-H) \sum_{i=1}^{\log_2 (NK-H)} \frac{(2^{2^{i-1}}-1)^2}{2^i} + 2^{NK-H} + H - 1 \\ &\leq 2^{NK-H+1} + \left(\log_2 \frac{NK-H}{2}\right) 2^{\frac{NK-H}{2}+1} + H - 1 \end{aligned} \tag{3.1.54}$$

where $H = \mathrm{Ham}(G)$.

Next we consider how many additions, C_D, are required to calculate all the distinct branch metric absolute differences, $\left|\underline{b}^{(-)}\right|$. Recall from (3.1.13) that

$$b_{n,k}^{(+)} = \begin{cases} 0 & \text{if } g_k = 0 \\ (-1)^{y_{n,k}^{(0)}} r_{n,k} & \text{if } g_k = 1 \end{cases} \tag{3.1.55}$$

Therefore each distinct $\underline{b}^{(-)}$ is generated by the following varying sum

$$\sum_{n,k|g_{n,k}=1} b_{n,k}^{(-)} = \sum_{n,k|g_{n,k}=1} \begin{cases} r_{n,k} & \text{if } y_{n,k}^{(0)} = 0 \\ -r_{n,k} & \text{if } y_{n,k}^{(0)} = 1 \end{cases} \tag{3.1.56}$$

in which there are $\mathrm{Ham}(G)$ terms in the summation.

Note that for every positive sum we compute, we also compute the negative sum which corresponds to the opposite ideal outputs. Therefore we fix a single $y_{n,k}^{(0)}$ and half the number of distinct summations we need to compute, leaving $2^{\mathrm{Ham}(G)-1}$ distinct summations. Since we actually need $\left|\underline{b}^{(-)}\right|$, taking the absolute value of each of the $2^{\mathrm{Ham}(G)-1}$ distinct summations will give all the distinct absolute differences. The number of absolute values required is therefore

$$C_{DA} = 2^{\mathrm{Ham}(G)-1} \tag{3.1.57}$$

For simplicity, suppose that $T = \mathrm{Ham}(G) - 1$ is one greater than a power of 2. The using the symmetry above, we can form all possible summations using a binary tree sharing all intermediate nodes in a similar manner to (2.5.6). This requires C_{DV} additions

$$C_{DV} = T \sum_{i=1}^{\log_2 T} \frac{2^{2^i}}{2^i} \tag{3.1.58}$$

Reapplying the upper bound determined in (2.5.8), we can bound C_{DV} as follows

$$\begin{aligned} C_{DV} &= T \sum_{i=1}^{\log_2 T} \frac{2^{2^i}}{2^i} \\ &\leq 2^T + \left(\log_2 \frac{T}{2}\right) 2^{\frac{T}{2}+1} \end{aligned} \tag{3.1.59}$$

Therefore the total addition count for calculating all the possible distinct

branch metric absolute differences is

$$\begin{aligned} C_D &\leq (H-1) \sum_{i=1}^{\log_2(H-1)} \frac{(2^{2^{i-1}}-1)^2}{2^i} + 2^{H-1} \\ &\leq 2^H + \left(\log_2 \frac{H-1}{2}\right) 2^{\frac{H-1}{2}+1} \end{aligned} \tag{3.1.60}$$

where $H = \mathrm{Ham}(G)$.

Therefore the total number of additions required to compute the maximum of all the branch metrics between the sides of loops is bounded by

$$\begin{aligned} C &= C_S + C_D + C_M \\ &\leq 2^{2K-2} + 2^{NK-H+1} + 2^H + \left(\log_2 \frac{NK-H}{2}\right) 2^{\frac{NK-H}{2}+1} \\ &\quad + \left(\log_2 \frac{H-1}{2}\right) 2^{\frac{H-1}{2}+1} + H - 1 \end{aligned} \tag{3.1.61}$$

3.2 Communications Example

Consider the $K = 7$ rate $\frac{1}{2}$ convolutional code octal (171,133)

$$\begin{aligned} y_k^{(0)} &= x_k \oplus x_{k-1} \oplus x_{k-2} \oplus x_{k-3} \oplus x_{k-6} \\ y_k^{(1)} &= x_k \oplus x_{k-2} \oplus x_{k-3} \oplus x_{k-5} \oplus x_{k-6} \end{aligned} \tag{3.2.1}$$

The generator matrix for this code is given by

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{3.2.2}$$

According to (2.5.9) the number of operations per K-T time step to calculate the branch metrics and eliminate the loops is bounded by

$$13908 \leq C_B + C_L \leq 14905 \tag{3.2.3}$$

According to (3.1.61) the number of operations per K-T time step to eliminate the loops using loop invariants is bounded by

$$C \leq 5258 \tag{3.2.4}$$

Therefore the use of invariants in the branch metric unit allows a significant reduction in complexity.

Chapter 4

Magnetic Channel

4.1 Read Channel Noise Model

We can model the magnetic channel as a sampled read-back of a superposition of isolated transitions [24].

$$r_l = \sum_k y_k h(lT - kT + j_k, w + w_k) + n_l \tag{4.1.1}$$

where $y_k \in \{-1, 0, +1\}$ is the transition sequence defined by

$$y_k = x_k - x_{k-1} \tag{4.1.2}$$

where $x_k \in \{0, 1\}$ is the original sequence.

$h(t, w)$ is the magnetic head response to an isolated transition, and j_k, w_k and n_l are independent random variables with standard deviations σ_j, σ_w

and σ_{wn} respectively.

The random variable j_k is referred to as position jitter [25, 26] as it changes the position in time of the isolated transition, w_k is referred to as pulse jitter as it changes the width of the isolated transition [25,27,28], and n_l is referred to as electronics noise [29].

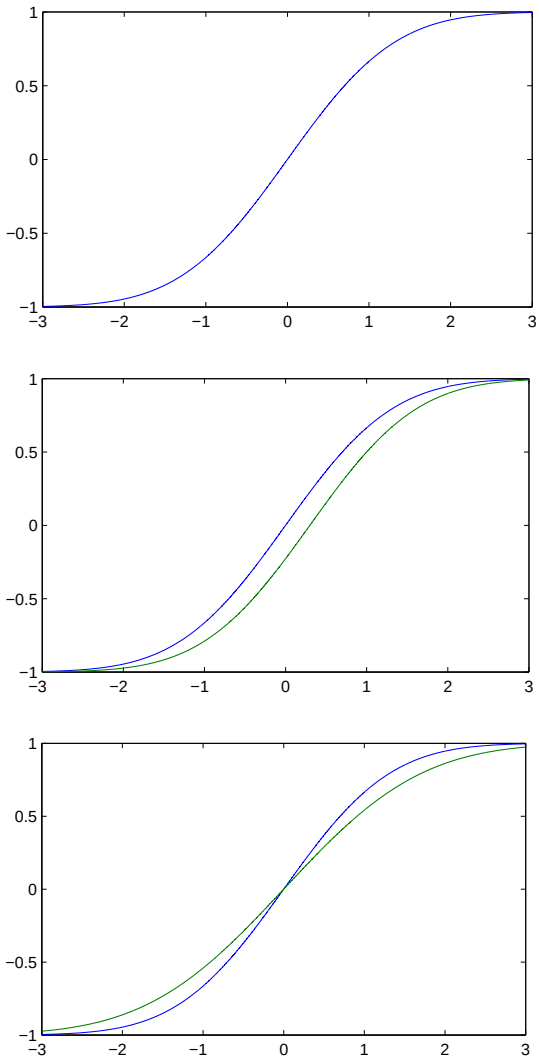


Figure 4.1: Isolated transition, isolated transition with position jitter and isolated transition with phase jitter, $T_{50} = 1.4$.

These random variables reflect the changing magnetic properties of the disk and the fluctuations in the field gradient of the writing head, that result in changes to the shape of each isolated transition [30].

Experimentally we can determine the shape of an isolated transition, $h(t, w)$, the pulse width of an isolated transition, w, the strength of the noise relative to the clean signal (signal to noise ratio), SNR, and the relative strengths of each noise component (electronics, position jitter and pulse jitter), which we denote α and λ as defined in (4.2.3) and (4.2.4).

From these experimental parameters, we can determine the distribution and standard deviation of j_k, w_k and n_l, which completely define the noise model.

4.2 Determining model parameters

Given the noise model from (4.1.1), we need to determine the model parameters from experimental data such as the energy of the transition response $h(t, w)$ [31].

Let us define E_S to be the total energy of the clean signal, E_E the total energy of the electronics noise and E_M to be the total energy of the media noise, where media noise consists of the position jitter and phase jitter components with total energies E_J and E_W, such that

$$E_M = E_J + E_W \tag{4.2.1}$$

The signal to noise ratio, SNR, is the strength of the total noise relative to

the clean signal, given in decibels.

$$\frac{E_S}{E_E + E_M} = 10^{\frac{SNR}{10}} \tag{4.2.2}$$

Let α define the fraction of total noise energy due to media noise

$$\alpha = \frac{E_M}{E_E + E_M} \tag{4.2.3}$$

and λ define the fraction of media noise due to position jitter

$$\lambda = \frac{E_J}{E_M} \tag{4.2.4}$$

We can then express each noise component in terms of the total noise energy of the clean signal, E_S, the signal to noise ratio, SNR, and the noise mixture parameters, α and λ.

$$\begin{aligned} E_E &= (1-\alpha)E_S 10^{\frac{-SNR}{10}} \\ E_J &= \lambda\alpha E_S 10^{\frac{-SNR}{10}} \\ E_W &= (1-\lambda)\alpha E_S 10^{\frac{-SNR}{10}} \end{aligned} \tag{4.2.5}$$

We must now derive an expression for E_S for the chosen isolated transition, and expressions for E_E, E_J and E_W in terms of their corresponding variances [32].

4.2.1 Clean signal energy and representation as linear ISI

Consider the expression for the clean signal I using equation (4.1.1) with noise set to zero

$$I_l = \sum_k y_k h(lT - kT, w) \tag{4.2.6}$$

We now show how the clean signal can be written in the form of a generalised partial response target response.

First we substitute the transitions $y_k \in \{-1, 0, +1\}$ with the original sequence $x_k \in \{0, 1\}$ using (4.1.2)

$$I_l = \sum_k (x_k - x_{k-1}) h(lT - kT, w) \tag{4.2.7}$$

Let us assume the original sequence has only a finite number of non-zero terms, then there exists $L > 0$ such that $x_k = 0$ for $|k| > L$. This allows us to rewrite (4.2.7) as a finite sum.

$$\begin{aligned} I_l &= \sum_{k=-L}^{L} x_k h(lT - kT, w) - \sum_{k=-L+1}^{L+1} x_{k-1} h(lT - kT, w) \\ &= \sum_{k=-L}^{L} x_k h(lT - kT, w) - \sum_{k=-L}^{L} x_k h(lT - (k+1)T, w) \\ &= \sum_{k=-L}^{L} x_k (h(lT - kT, w) - h(lT - (k+1)T, w)) \end{aligned} \tag{4.2.8}$$

Now use a change of variables $k \leftarrow l - k$ to obtain

$$I_l = \sum_{k=l-L}^{l+L} x_{l-k}(h(kT, w) - h((k-1)T, w)) \tag{4.2.9}$$

For convenience, since $x_k = 0$ for $|k| > L$ we can rewrite (4.2.9) as an infinite sum (of finitely many non-zero terms) to achieve our final expression for I

$$\begin{aligned} I_l &= \sum_k x_{l-k}(h(kT, w) - h((k-1)T, w)) \\ &= \sum_k g_k x_{l-k} \end{aligned} \tag{4.2.10}$$

where

$$g_k = h(kT, w) - h((k-1)T, w) \tag{4.2.11}$$

We can think of $G = \{g_k\}$ as the natural target or generalised partial response target.

We can now check that when there is no ISI, this target agrees with intuition. Suppose that at density w_0 the sampled isolated transition is given by (i.e. there is no ISI)

$$h(kT, w_0) = \begin{cases} -1 & \text{if } k < 0 \\ 0 & \text{if } k = 0 \\ +1 & \text{if } k > 0 \end{cases} \tag{4.2.12}$$

Using (4.2.11) we can compute the natural target

$$g_k = h(kT, w_0) - h((k-1)T, w_0) = \begin{cases} 0 & \text{if } k < 0 \\ 1 & \text{if } k = 0 \\ 1 & \text{if } k = 1 \\ 0 & \text{if } k > 1 \end{cases} \tag{4.2.13}$$

So as we expect, the generalised partial response target for densities low enough such that there is no ISI is $[1, 1]$.

Also, consider density w_1 where $N = 2$.

$$h(kT, w_1) = \begin{cases} -1 & \text{if } k < -1 \\ -p & \text{if } k = -1 \\ 0 & \text{if } k = 0 \\ +p & \text{if } k = 1 \\ +1 & \text{if } k > 1 \end{cases} \tag{4.2.14}$$

Using (4.2.11) we can compute the natural target

$$g_k = h(kT, w_1) - h((k-1)T, w_1) = \begin{cases} 0 & \text{if } k < -1 \\ 1-p & \text{if } k = -1 \\ p & \text{if } k = 0 \\ p & \text{if } k = 1 \\ 1-p & \text{if } k = 2 \\ 0 & \text{if } k > 2 \end{cases} \tag{4.2.15}$$

So at densities where $N = 2$ the generalised partial response target is $[1 - p, p, p, 1 - p]$.

We can clearly see how the length of the generalised partial response target is directly related to the integer N which increases with density.

In general for the perpendicular channel, isolated transitions have the following property

$$\begin{aligned} \lim_{t \to +\infty} h(t, w) &= +1 \\ \lim_{t \to -\infty} h(t, w) &= -1 \end{aligned} \tag{4.2.16}$$

For example, the following functions all satisfy the above criteria

$$\begin{aligned} h_1(t,w) &= \operatorname{erf}\left(\frac{t}{w}\right) \\ h_2(t,w) &= \tanh\left(\frac{t}{w}\right) \\ h_3(t,w) &= \tan^{-1}\left(\frac{t}{w}\right) \\ h_4(t,w) &= \begin{cases} -1 & \text{if } t \le -N \\ f(t,w) & \text{if } -N < t < N \\ 1 & \text{if } t \ge N \end{cases} \end{aligned} \tag{4.2.17}$$

For perpendicular recording, tangent hyperbolic, arctangent and error functions have all been used for the isolated transition shape [33].

Since the value of the isolated transition approaches a constant away from the origin, the differences between the values of consecutive samples become effectively zero sufficiently far from the origin.

In particular considering (4.2.16), given $\varepsilon > 0$, there exists $N > 0$ such that $|h(kT,w) - 1| < \varepsilon$ and $|h(-kT,w) + 1| < \varepsilon$ for all $k \ge N$.

Therefore if $k > N$

$$\begin{aligned} |g_k| &= |h(kT,w) - h((k-1)T,w)| \\ &= |(h(kT,w) - 1) - (h((k-1)T,w) - 1)| \\ &\le |h(kT,w) - 1| + |h((k-1)T,w) - 1| \\ &< 2\varepsilon \end{aligned} \tag{4.2.18}$$

and if $k \leq -N$

$$\begin{aligned} |g_k| &= |h(kT, w) - h((k-1)T, w)| \\ &= |(h(kT, w) + 1) - (h((k-1)T, w) + 1)| \\ &\leq |h(kT, w) + 1| + |h((k-1)T, w) + 1| \\ &< 2\varepsilon \end{aligned} \tag{4.2.19}$$

Hence we have a generalised partial response target $G = [g_{-N+1}, \ldots, g_N]$ of length $2N$.

4.2.2 Media Noise

Having determined the strength of the clean signal, we must now determine the relative strength of the media and electronics noise to determine the correct noise mixture.

It should be noted that in current systems, media noise dominates electronics noise [34].

With I given in the form (4.2.10) it is easy to compute E_S the energy of the clean signal by computing the variance of I

$$E_S = \mathbb{E}[I_l^2] - \mathbb{E}[I_l]^2 \tag{4.2.20}$$

But $\mathbb{E}[I_l]^2$ is very easy to compute since $\mathbb{E}[x_i] = \frac{1}{2}$

$$\begin{aligned}\mathbb{E}[I_l]^2 &= \mathbb{E}\left[\sum_k g_k x_{l-k}\right]^2 \\ &= \left(\sum_k g_k \mathbb{E}[x_{l-k}]\right)^2 \\ &= \frac{1}{4}\left(\sum_k g_k\right)^2 \\ &= \frac{1}{4}\sum_k \sum_j g_k g_j\end{aligned} \tag{4.2.21}$$

and we can compute $\mathbb{E}[I_l^2]$ using $\mathbb{E}[x_i^2] = \frac{1}{2}$ and $\mathbb{E}[x_i x_j] = \frac{1}{4}$ for $i \neq j$

$$\begin{aligned}\mathbb{E}[I_l^2] &= \mathbb{E}\left[\left(\sum_k g_k x_{l-k}\right)^2\right] \\ &= \mathbb{E}\left[\sum_k \sum_j g_k g_j x_{l-k} x_{l-j}\right] \\ &= \sum_k \sum_j g_k g_j \mathbb{E}[x_{l-k} x_{l-j}] \\ &= \frac{1}{2}\sum_k g_k^2 + \frac{1}{4}\sum_k \sum_{j\neq k} g_k g_j \\ &= \frac{1}{4}\sum_k g_k^2 + \frac{1}{4}\sum_k \sum_j g_k g_j\end{aligned} \tag{4.2.22}$$

Therefore E_S is given by

$$\begin{aligned}E_S &= \mathbb{E}[I_l^2] - \mathbb{E}[I_l]^2 \\ &= \frac{1}{4}\sum_k g_k^2\end{aligned} \tag{4.2.23}$$

So now we have an expression for E_S which we can compute numerically (as long as we choose N big enough).

$$E_S = \frac{1}{4}\sum_k [h(kT, w) - h((k-1)T, w)]^2 \tag{4.2.24}$$

Note that this can also be thought of as $\frac{1}{4}$ of the total energy of a dibit response

Now we know E_S, we can compute E_E, E_J and E_W using equations (4.2.5). All that remains is to derive equations for the variances of the random variables given the respective energies.

$$E_E = \mathbb{E}[n_l^2] = \sigma_E^2 \tag{4.2.25}$$

Total media noise can be expressed by removing the clean signal from the noise signal (without white noise) - with a first-order Taylor expansion it can then be expressed as a distinct sum of contributions from position jitter and pulse jitter.

$$\begin{aligned} N_M &= \sum_k y_k h(-kT + j_k, w + w_k) - \sum_k y_k h(-kT, w) \\ &\approx \sum_k y_k \frac{\partial}{\partial t} h(-kT, w) j_k + \sum_k y_k \frac{\partial}{\partial w} h(-kT, w) w_k \end{aligned} \tag{4.2.26}$$

Given that derivatives of h look like a longitudinal pulse - they quickly decay to zero either side of the peak, we don't need to worry about boundary conditions when computing these sums. Now we can derive expressions for

E_J and E_W

$$\begin{aligned}
E_M &= \mathbb{E}\left[\left[\sum_k y_k \frac{\partial}{\partial t} h(-kT, w) j_k + \sum_k y_k \frac{\partial}{\partial w} h(-kT, w) w_k\right]^2\right] \\
E_M &= \mathbb{E}\left[\left[\sum_k y_k \frac{\partial}{\partial t} h(-kT, w) j_k\right]^2\right] + \mathbb{E}\left[\left[\sum_k y_k \frac{\partial}{\partial w} h(-kT, w) w_k\right]^2\right] \\
&= E_J + E_W \\
E_J &= \sum_k \mathbb{E}[y_k^2]\mathbb{E}[j_k^2] \left(\frac{\partial}{\partial t} h(-kT, w)\right)^2 \\
&= \frac{1}{2}\sigma_J^2 \sum_k \left(\frac{\partial}{\partial t} h(-kT, w)\right)^2 \\
E_W &= \frac{1}{2}\sigma_W^2 \sum_k \left(\frac{\partial}{\partial w} h(-kT, w)\right)^2
\end{aligned} \tag{4.2.27}$$

Computing these derivatives we find

$$\sum_k \left(\frac{\partial}{\partial t} h(-kT, w)\right)^2 = \quad h_t = \frac{4}{\pi w^2} \sum_k e^{-2(kT/w)^2} \tag{4.2.28}$$

$$\sum_k \left(\frac{\partial}{\partial w} h(-kT, w)\right)^2 = \quad h_w = \frac{4}{\pi w^4} \sum_k (kT)^2 e^{-2(kT/w)^2} \tag{4.2.29}$$

We now have a complete system of equations to compute variances of the three noise sources

$$\sigma_E^2 = \quad (1-\alpha) E_S 10^{-SNR/10} \tag{4.2.30}$$

$$\sigma_J^2 = \quad 2\lambda\alpha \frac{E_S}{h_t} 10^{-SNR/10} \tag{4.2.31}$$

$$\sigma_W^2 = \quad 2(1-\lambda)\alpha \frac{E_S}{h_w} 10^{-SNR/10} \tag{4.2.32}$$

where h_t is defined in (4.2.28), h_w defined in (4.2.29), E_S defined in (4.2.24). At this point we have just computed variances for the jitter random variables. If we want the jitter random variables to be normally distributed, we can just generate Gaussian random variables with the variances computed in (4.2.31) and (4.2.32)

$$j \sim N(0, \sigma_J^2) \qquad w \sim N(0, \sigma_W^2) \qquad n \sim N(0, \sigma_E^2) \tag{4.2.33}$$

We could also model jitter with uniform random variables. If we use a uniform distribution $U(-a, a)$ whose probability density function is constant over the interval $[-a, +a]$ then the variance of this distribution is $a^2/3$ and we can generate uniformly distributed jitter random variables, along with the Gaussian electronics noise, as follows

$$j \sim U(-\sqrt{3}\sigma_J, +\sqrt{3}\sigma_J) \qquad w \sim U(-\sqrt{3}\sigma_W, +\sqrt{3}\sigma_W) \qquad n \sim N(0, \sigma_E^2) \tag{4.2.34}$$

4.2.3 Isolated Transition - Error Function

The most common model for the isolated transition shape in perpendicular recording is the error function [35].

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt \tag{4.2.35}$$

Let $h(t, w)$ denote the response to an isolated transition

$$h(t, w) = \text{erf}\left(\frac{t}{w}\right) \tag{4.2.36}$$

We choose w such that the pulse width at the 50% point between the baseline and the peak is $t = T_{50}$ [36], as shown in figure 4.2.

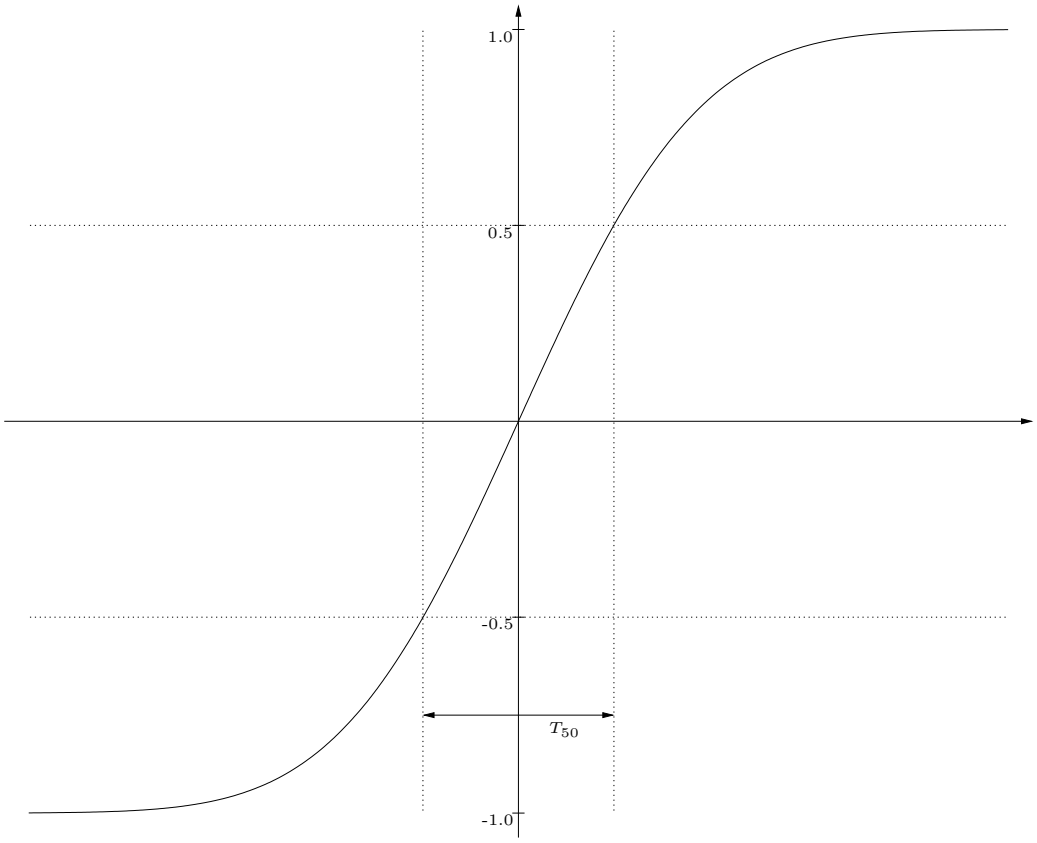


Figure 4.2: Measuring T_{50}.

Therefore we must solve

$$h(T_{50}, w) = \text{erf}\left(\frac{\frac{1}{2}T_{50}}{w}\right) = 0.5 \tag{4.2.37}$$

which has the solution

$$w \approx 1.048358\, T_{50} \tag{4.2.38}$$

Hence we can express the isolated transition in terms of T_{50} as

$$h(t, T_{50}) \approx \operatorname{erf}\left(\frac{t}{1.048358\, T_{50}}\right) \tag{4.2.39}$$

4.2.4 Isolated Transition - Hyperbolic Tangent

Another common isolated transition in perpendicular recording is hyperbolic tangent [33].

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{4.2.40}$$

Let $h(t, w)$ denote the response to an isolated transition

$$h(t, w) = \tanh\left(\frac{t}{w}\right) \tag{4.2.41}$$

We choose w such that the pulse width at the 50% point between the baseline and the peak is $t = T_{50}$, as shown in figure 4.2. Therefore we must solve

$$h(T_{50}, w) = \tanh\left(\frac{\frac{1}{2}T_{50}}{w}\right) = 0.5 \tag{4.2.42}$$

which has the solution

$$w \approx \frac{1}{\ln 3} T_{50} \tag{4.2.43}$$

Hence we can express the isolated transition in terms of T_{50} as

$$h(t, T_{50}) \approx \tanh\left(\frac{t \ln 3}{T_{50}}\right) \tag{4.2.44}$$

Chapter 5

Data Dependent Detectors

The magnetic channel noise model we discussed in the previous chapter described a channel in which noise is both correlated and data dependent [24].

$$r_l = \sum_k y_k h(lT - kT + j_k, w + w_k) + n_l \tag{5.0.1}$$

Regular Viterbi detectors [37] are only optimal when the noise is white Gaussian, since they do not take into account correlations between noise samples.

Noise prediction [38, 39] achieves improved performance by estimating the current noise sample from the previous noise samples, then subtracting the estimate from the actual noise sample, effectively whitening the noise by removing the correlations.

In high density magnetic recording channels, data dependent media noise exists [11, 40] due to changing magnetic properties of the disk and the fluctuations in the field gradient [26, 41].

Further improvements to the performance can be achieved by considering the data dependence of the noise [11, 42]. Data dependent detectors were first proposed in [43, 44] and later on was treated more generally in [45].

The aim of this chapter is to discuss various Viterbi detector implementations which assume noise models that approximate the above general model, then to minimise the Kullback-Leibler divergence [46] between the approximate and ideal noise models to determine the noise model parameters and provide simulation results to compare the various implementations.

The first implementation we shall consider assumes that the noise is auto-regressive with correlation length L and data-independent, i.e. noise samples are correlated with the previous L noise sample, but those correlations do not depend on the original data.

5.1 Auto-Regressive Noise Viterbi Detector

Theorem 5.1.1. Suppose a binary sequence $\underline{x} \in \{0,1\}^N$ which is encoded with a non-recursive convolutional code describing inter-symbol interference (ISI) with impulse response $\{g_0, \ldots, g_I\}$,

$$y_i = \sum_{k=0}^{I} g_k x_{i-k} \tag{5.1.1}$$

is transmitted over an auto-regressive Gaussian noise channel,

$$r_i = y_i + n_i \tag{5.1.2}$$

where

$$n_i = \sigma\omega_i + \sum_{l=1}^{L} b_l n_{i-l} \tag{5.1.3}$$

and where $\omega_i \sim N(0,1)$. Then given the received sequence $\underline{r} \in \mathbb{R}^N$, the original unencoded sequence $\underline{\hat{x}} \in \{0,1\}^N$ transmitted with maximum likelihood is given by

$$\underline{\hat{x}} = \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} BM_i(\underline{n}) \right\} \tag{5.1.4}$$

where the branch metric $BM_i(\underline{n})$ is given by

$$BM_i(\underline{n}) = \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \tag{5.1.5}$$

Proof. Since $\omega_i \sim N(0,1)$ are independent variables, we have the following expression for the conditional probability of a given noise sequence

$$\Pr(\underline{r} \mid \underline{x}) = \prod_{i=0}^{N-1} \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2} \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \right] \tag{5.1.6}$$

From (1.2.5), in order to find the maximally likely sequence, we need to determine

$$\begin{aligned} \underline{\hat{x}} &= \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmax}} \{\Pr(\underline{r} \mid \underline{x})\} \\ &= \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \{-2\ln \Pr(\underline{r} \mid \underline{x})\} \\ &= \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ -2\ln \left\{ \prod_{i=0}^{N-1} \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2} \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \right] \right\} \right\} \end{aligned} \tag{5.1.7}$$

Simplifying the logarithms and exponentials gives

$$\underline{\hat{x}} = \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} \ln 2\pi\sigma^2 + \frac{1}{\sigma^2} \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \right\} \tag{5.1.8}$$

which is equivalent to

$$\underline{\hat{x}} = \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \right\} \tag{5.1.9}$$

since the terms in σ are common to all paths and therefore don't effect the argument of the minimisation. □

5.1.1 Computation of Noise Parameters

Suppose the noise source is unknown and has probability density function $P(\underline{n}_i)$. We need to choose parameters $\underline{b}$ such that (5.1.3) matches the unknown noise model $P(\underline{n}_i)$ as closely as possible. Therefore we need to choose $\underline{b}$ such that $D_{KL}(P \mid P^{(0)})$ is minimal, where $P^{(0)}$ is given by (5.1.6).

By definition

$$\begin{aligned} D_{KL}(P \mid P^{(0)}) &= \int P(\underline{n}) \ln \frac{P(\underline{n})}{P^{(0)}(\underline{n})} \prod_{k=1}^{N} dn_k \\ &= \int P(\underline{n}) \ln P(\underline{n}) \prod_{k=1}^{N} dn_k - \int P(\underline{n}) \ln P^{(0)}(\underline{n}) \prod_{k=1}^{N} dn_k \end{aligned} \tag{5.1.10}$$

We obtain optimality when

$$\frac{\partial}{\partial \underline{b}} D_{KL}(P \mid P^{(0)}) = 0 \tag{5.1.11}$$

We can express $P^{(0)}$ as

$$P^{(0)} = e^{-\prod_{k=1}^{N} BM_k} \tag{5.1.12}$$

Therefore

$$\begin{aligned}
\frac{\partial}{\partial \underline{b}} D_{KL}(P \mid P^{(0)}) &= \sum_{k=1}^{N} \frac{\partial}{\partial \underline{b}} \int P(\underline{n}) \ln P^{(0)}(\underline{n}) \prod_{k=1}^{N} dn_k \\
&= -\sum_{k=1}^{N} \frac{\partial}{\partial \underline{b}} \int P(\underline{n}) BM_k(\underline{n}) \prod_{l=1}^{N} dn_l \\
&= -\sum_{k=1}^{N} \frac{\partial}{\partial \underline{b}} \mathbb{E}_P[BM_k(\underline{n})] \\
&= -N \frac{\partial}{\partial \underline{b}} \mathbb{E}_P[BM_k(\underline{n})]
\end{aligned} \tag{5.1.13}$$

Hence finding $\underline{b}$ is equivalent to $\mathbb{E}$-branch metric minimisation.

From (5.1.5) we see the branch metric is given by

$$BM_k(\underline{n}) = \left(n_k - \sum_{l=1}^{L} b_l n_{k-l} \right)^2 \tag{5.1.14}$$

Setting the partial derivatives to zero shows that for $m = 1, \ldots, L$

$$\begin{aligned}
0 = -\frac{1}{2} \frac{\partial}{\partial b_m} \mathbb{E}_P[BM_k(\underline{n})] &= \mathbb{E}_P \left[n_{k-m} \left(n_k - \sum_{l=1}^{L} b_l n_{k-l} \right) \right] \\
&= \mathbb{E}_P \left[n_{k-m} n_k \right] - \sum_{l=1}^{L} b_l \mathbb{E}_P \left[n_{k-m} n_{k-l} \right]
\end{aligned} \tag{5.1.15}$$

Hence

$$\underline{b} = \mathbf{C}^{-1}\underline{v} \tag{5.1.16}$$

where $\mathbf{C}$ is the correlation matrix [1]

$$\mathbf{C} = \begin{bmatrix} \mathbb{E}_P[n_{k-1}^2] & \mathbb{E}_P[n_{k-1}n_{k-2}] & \dots & \mathbb{E}_P[n_{k-1}n_{k-L}] \\ \mathbb{E}_P[n_{k-2}n_{k-1}] & \mathbb{E}_P[n_{k-2}^2] & \dots & \mathbb{E}_P[n_{k-2}n_{k-L}] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}_P[n_{k-L}n_{k-1}] & \mathbb{E}_P[n_{k-L}n_{k-2}] & \dots & \mathbb{E}_P[n_{k-L}^2] \end{bmatrix} \tag{5.1.17}$$

and $\underline{v}$ is the vector

$$\underline{v} = \begin{bmatrix} \mathbb{E}_P[n_k n_{k-1}] \\ \mathbb{E}_P[n_k n_{k-2}] \\ \vdots \\ \mathbb{E}_P[n_k n_{k-L}] \end{bmatrix} \tag{5.1.18}$$

5.1.2 Simulation Results

The following plot shows the relative performance of a data independent auto-regressive detector compared to a white noise detector.

The choice of target for the white noise detector was determined using the minimum mean squared error (MMSE) criteria, whilst the target for the auto-regressive detector was chosen as the best performing target obtained from a target search.

[1] The proper calculation of $\mathbf{C}^{-1}$ should be based on the singular value decomposition (SVD) and generalised inverse.

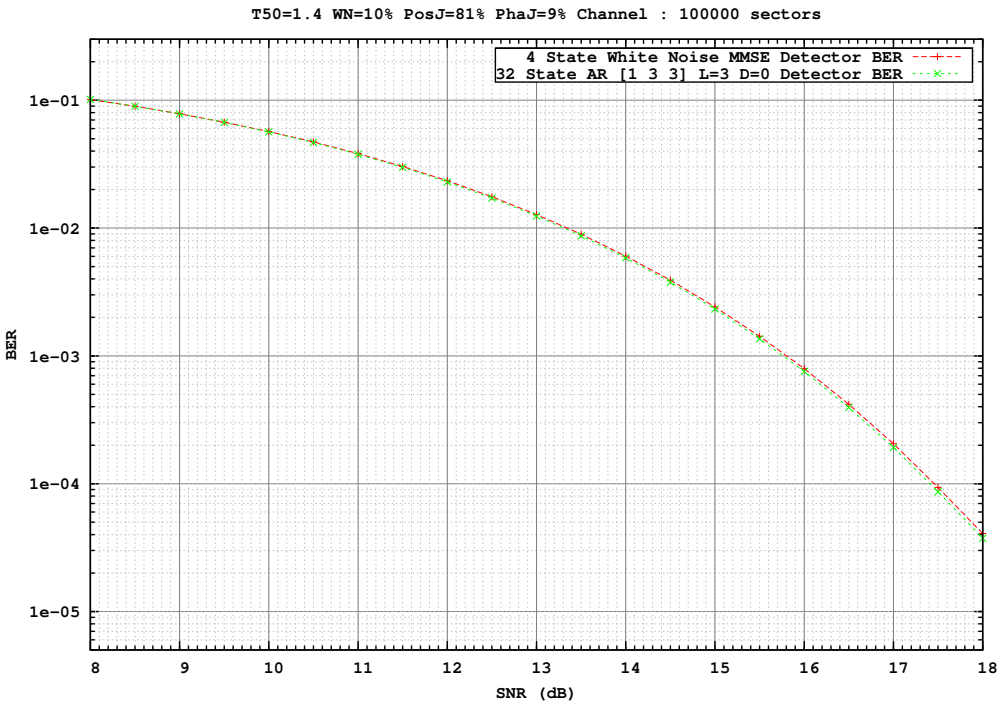


Figure 5.1: Comparison of white noise MMSE detector and data independent auto-regressive detector.

5.1.3 Limitations

The branch metric for the data independent auto-regressive detector is given by

$$BM_i = \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \tag{5.1.19}$$

where

$$n_j = r_j - y_j = r_j - \sum_{k=0}^{I} g_k x_{j-k} \tag{5.1.20}$$

Hence n_j depends on $\{x_{j-I}, \ldots, x_j\}$ and BM_i depends on $\{n_{i-L}, \ldots, n_i\}$. Therefore BM_i depends on $\{x_{i-I-L}, \ldots, x_i\}$.

The Viterbi detector therefore requires 2^{I+L} states in order to keep track of the $I + L$ previous terms from the original sequence.

Note that this is 2^L times more states than the equivalent white noise Viterbi detector. Hence the size of the auto-regressive detector increases exponentially with the correlation length L.

Also figure 5.1 shows that the performance gain relative to the 2^I state white noise detector is minimal.

5.1.4 Equivalence To White Noise Detector

Note that figure 5.1 suggests that data independent auto-regressive detectors perform comparably to white noise detectors when optimal targets are chosen for both detectors.

To understand this, let us rearrange the branch metric from (5.1.19)

$$\begin{aligned} BM_i &= \left(n_i - \sum_{l=1}^{L} b_l n_{i-l} \right)^2 \\ &= \left((r_i - y_i) - \sum_{l=1}^{L} b_l (r_{i-l} - y_{i-l}) \right)^2 \\ &= \left(\left(r_i - \sum_{l=1}^{L} b_l r_{i-l} \right) - \left(y_i - \sum_{l=1}^{L} b_l y_{i-l} \right) \right)^2 \\ &= (R_i - Y_i)^2 \end{aligned} \tag{5.1.21}$$

where

$$\begin{aligned} R_i &= r_i - \sum_{l=1}^{L} b_l r_{i-l} = \underline{r} * \{1, -\underline{b}\} \\ Y_i &= y_i - \sum_{l=1}^{L} b_l y_{i-l} = \underline{y} * \{1, -\underline{b}\} = \underline{x} * (\underline{g} * \{1, -\underline{b}\}) \end{aligned} \tag{5.1.22}$$

Therefore data independent auto-regressive detectors with target length $I+1$ and correlation length L are equivalent to white noise detectors with target length $I+L+1$ where the target is determined by the convolution $\underline{g}*\{1, -\underline{b}\}$ and the received signal is additionally equalised by applying a finite impulse response filter with tap coefficients $\{1, -\underline{b}\}$.

5.2 Data Dependent Auto-Regressive Noise Viterbi Detector

A refinement to the auto-regressive model is to assume that in addition being conditionally Gaussian and Markov, the noise correlations depend on the original data [11].

Let $x_0^i = \{x_0, x_1, \ldots, x_i\}$ represent a sub-sequence of the original data sequence. Note that we assume elements of the original data sequence are independent identically distributed random variables taking values 0 and 1, with probability $\frac{1}{2}$.

We define the sequence of noise signals $\{n_0, n_1, \ldots n_i\}$ to be

$$n_i = \mu(x_{i-D}^i) + \sigma(x_{i-D}^i)\omega_i + \sum_{k=1}^{L} b_k(x_{i-D}^i)n_{i-k} \tag{5.2.1}$$

where x_{i-D}^i is the data pattern and $\omega_i \sim N(0,1)$ are independent identically distributed standard Gaussian random variables.

Theorem 5.2.1. Suppose a binary sequence $\underline{x} \in \{0,1\}^N$ which is encoded with a non-recursive convolutional code describing inter-symbol interference (ISI) with impulse response $\{g_0, \ldots, g_I\}$,

$$y_i = \sum_{k=0}^{I} g_k x_{i-k} \tag{5.2.2}$$

is transmitted over an auto-regressive Gaussian noise channel,

$$r_i = y_i + n_i \tag{5.2.3}$$

where n_i is defined by (5.2.1) and where $\omega_i \sim N(0,1)$. Then given the received sequence $\underline{r} \in \mathbb{R}^N$, the original unencoded sequence $\hat{\underline{x}} \in \{0,1\}^N$ transmitted with maximum likelihood is given by

$$\hat{\underline{x}} = \underset{\underline{x}\in\{0,1\}^N}{\operatorname{argmin}} \left\{ \sum_{i=0}^{N-1} BM_i(x_{i-D}^i, n_{i-L}^i) \right\} \tag{5.2.4}$$

where the branch metric $BM_i(x_{i-D}^i, n_{i-L}^i)$ is given by

$$BM_i(x_{i-D}^i, n_{i-L}^i) = \ln(\sigma(x_{i-D}^i)) + \frac{1}{\sigma^2(x_{i-D}^i)} \left(n_i - \sum_{l=1}^{L} b_l(x_{i-D}^i)n_{i-l} - \mu(x_{i-D}^i) \right)^2 \tag{5.2.5}$$

Proof. Using the fact that ω_i are independent normal variables, the conditional probability of a given noise sequence is given by

$$P(n_0^N \mid x_0^N) = \prod_{i=0}^{N} \sqrt{\frac{1}{2\pi\sigma^2(x_{i-D}^i)}} e^{-\frac{1}{2\sigma^2(x_{i-D}^i)}\left(n_i - \sum_{k=1}^{L} b_k(x_{i-D}^i) n_{i-k} - \mu(x_{i-D}^i)\right)^2}, \tag{5.2.6}$$

where we assumed that $n_k = 0, x_k = 0$ for $k < 0$, see [11].

Taking the natural logarithm of (5.2.6) one finds that

$$-2\ln\left(P(n_0^i \mid x_0^i)\right) = \sum_{k=0}^{i} BM(x_{i-D}^i, n_{i-L}^i), \tag{5.2.7}$$

where

$$BM(x_{i-D}^i, n_{i-L}^i) = 2\ln(\sigma(x_{i-D}^i)) + \frac{1}{\sigma^2(x_{i-D}^i)} \left(n_i - \sum_{k=1}^{L} b_k(x_{i-D}^i) n_{i-k} - \mu(x_{i-D}^i)\right)^2 \tag{5.2.8}$$

is the branch metric of maximum likelihood detector matched to data dependent auto-regressive noise (5.2.1) with Markov length L and data dependent length M. □

5.2.1 Computation of Noise Parameters

Suppose the noise source is unknown and has probability density function $P(\underline{n}_i)$. We need to choose parameters $\sigma, \underline{b}, \mu$ such that (5.2.1) matches the unknown noise model $P(\underline{n}_i)$ as closely as possible. Therefore we need to choose $\sigma, \underline{b}, \mu$ such that

$$D_{KL}(P \mid P^{(0)}) \tag{5.2.9}$$

is minimal, where $P^{(0)}$ is given by (5.2.6).

By definition

$$\begin{aligned} D_{KL}(P \mid P^{(0)}) &= \int P(\underline{n}) \ln \frac{P(\underline{n})}{P^{(0)}(\underline{n})} \prod_{k=1}^{N} dn_k \\ &= \int P(\underline{n}) \ln P(\underline{n}) \prod_{k=1}^{N} dn_k - \int P(\underline{n}) \ln P^{(0)}(\underline{n}) \prod_{k=1}^{N} dn_k \end{aligned} \tag{5.2.10}$$

We obtain optimality when

$$\frac{\partial}{\partial \{\sigma, \underline{b}, \mu\}} D_{KL}(P \mid P^{(0)}) = 0 \tag{5.2.11}$$

We can express $P^{(0)}$ as

$$P^{(0)} = e^{-\prod_{k=1}^{N} BM_k} \tag{5.2.12}$$

Therefore

$$\begin{aligned} \frac{\partial}{\partial \{\sigma, \underline{b}, \mu\}} D_{KL}(P \mid P^{(0)}) &= \sum_{k=1}^{N} \frac{\partial}{\partial \{\sigma, \underline{b}, \mu\}} \int P(\underline{n}) \ln P^{(0)}(\underline{n}) \prod_{k=1}^{N} dn_k \\ &= -\sum_{k=1}^{N} \frac{\partial}{\partial \{\sigma, \underline{b}, \mu\}} \int P(\underline{n}) BM_k(\underline{n}) \prod_{l=1}^{N} dn_l \\ &= -\sum_{k=1}^{N} \frac{\partial}{\partial \{\sigma, \underline{b}, \mu\}} \mathbb{E}_P[BM_k(\underline{n})] \\ &= -N \frac{\partial}{\partial \{\sigma, \underline{b}, \mu\}} \mathbb{E}_P[BM_k(\underline{n})] \end{aligned} \tag{5.2.13}$$

Hence finding $\sigma, \underline{b}, \mu$ is equivalent to $\mathbb{E}$-branch metric minimisation.

From (5.2.8) we see the branch metric is given by

$$BM_k(\underline{n},\underline{p}) = 2\ln(\sigma(\underline{p})) + \frac{1}{\sigma^2(\underline{p})}\left(n_k - \sum_{l=1}^{L} b_l(\underline{p})n_{k-l} - \mu(\underline{p})\right)^2 \tag{5.2.14}$$

Hence the expected value of the branch metric is given by

$$\mathbb{E}_P[BM_k(\underline{n})] = \frac{1}{2^{D+1}}\sum_{p=0}^{2^{D+1}-1}\mathbb{E}_P[BM_k(\underline{n},\underline{p}) \mid p_k = p] \tag{5.2.15}$$

Therefore when we differentiate we find that

$$\frac{\partial}{\partial\left\{b_m,\sigma,\mu\right\}(\underline{p})}\mathbb{E}_P[BM_k(\underline{n})] = \frac{1}{2^{D+1}}\frac{\partial}{\partial\left\{b_m,\sigma,\mu\right\}(\underline{p})}\mathbb{E}_P[BM_k(\underline{n},\underline{p}) \mid p_k = p] \tag{5.2.16}$$

This allows us to calculate $b_m(\underline{p}), \sigma(\underline{p}), \mu(\underline{p})$ for each pattern $\underline{p}$ independently.

Setting the partial derivative w.r.t. the data dependent mean to zero gives

$$\begin{aligned} 0 &= -2^D\frac{\partial}{\partial\mu(\underline{p})}\mathbb{E}_P[BM_k(\underline{n})] \\ &= \mathbb{E}_P\left[\frac{1}{\sigma^2(\underline{p})}\left(n_k - \sum_{l=1}^{L} b_l(\underline{p})n_{k-l} - \mu(\underline{p})\right) \mid p_k = p\right] \\ &= \frac{1}{\sigma^2(\underline{p})}\left(\mathbb{E}_P\left[n_k \mid p_k = p\right] - \sum_{l=1}^{L} b_l(\underline{p})\mathbb{E}_P\left[n_{k-l} \mid p_k = p\right] - \mu(\underline{p})\right) \end{aligned} \tag{5.2.17}$$

Hence

$$\mu(\underline{p}) = \mathbb{E}_P\left[n_k \mid p_k = p\right] - \sum_{l=1}^{L} b_l(\underline{p})\mathbb{E}_P\left[n_{k-l} \mid p_k = p\right] \tag{5.2.18}$$

Then setting the partial derivatives w.r.t. $b_m(\underline{p})$ to zero shows that for $m =$

$1, \ldots, L$

$$
\begin{aligned}
0 &= -2^D \frac{\partial}{\partial b_m(\underline{p})} \mathbb{E}_P[BM_k(\underline{n})] \\
&= \mathbb{E}_P\left[\frac{1}{\sigma^2(\underline{p})} n_{k-m}\left(n_k - \sum_{l=1}^{L} b_l(\underline{p}) n_{k-l} - \mu(\underline{p})\right) \mid p_k = p\right] \\
&= \frac{1}{\sigma^2(\underline{p})}\Bigg(\mathbb{E}_P\left[n_{k-m}n_k \mid p_k = p\right] \\
&\quad - \sum_{l=1}^{L} b_l(\underline{p})\mathbb{E}_P\left[n_{k-m}n_{k-l} \mid p_k = p\right] - \mu(\underline{p})\mathbb{E}_P\left[n_{k-m} \mid p_k = p\right]\Bigg) \\
&= \frac{1}{\sigma^2(\underline{p})}\Bigg(\mathbb{E}_P\left[n_{k-m}n_k \mid p_k = p\right] - \sum_{l=1}^{L} b_l(\underline{p})\mathbb{E}_P\left[n_{k-m}n_{k-l} \mid p_k = p\right] \\
&\quad - \left(\mathbb{E}_P\left[n_k \mid p_k = p\right] - \sum_{l=1}^{L} b_l(\underline{p})\mathbb{E}_P\left[n_{k-l} \mid p_k = p\right]\right)\mathbb{E}_P\left[n_{k-m} \mid p_k = p\right]\Bigg) \\
&= \frac{1}{\sigma^2(\underline{p})}\Bigg(\left(\mathbb{E}_P\left[n_{k-m}n_k \mid p_k = p\right] - \mathbb{E}_P\left[n_k \mid p_k = p\right]\mathbb{E}_P\left[n_{k-m} \mid p_k = p\right]\right) \\
&\quad - \sum_{l=1}^{L} b_l(\underline{p})\left(\mathbb{E}_P\left[n_{k-m}n_{k-l} \mid p_k = p\right] - \mathbb{E}_P\left[n_{k-l} \mid p_k = p\right]\mathbb{E}_P\left[n_{k-m} \mid p_k = p\right]\right)\Bigg) \\
&= \frac{1}{\sigma^2(\underline{p})}\left(\mathbb{E}_P\left[\eta_{k-m}\eta_k \mid p_k = p\right] - \sum_{l=1}^{L} b_l(\underline{p})\mathbb{E}_P\left[\eta_{k-m}\eta_{k-l} \mid p_k = p\right]\right)
\end{aligned}
\tag{5.2.19}
$$

where

$$\eta_i = n_i - \mathbb{E}_P[n_i] \tag{5.2.20}$$

Hence

$$\underline{b}(\underline{p}) = \mathbf{C}(\underline{p})^{-1}\underline{v}(\underline{p}) \tag{5.2.21}$$

where $\mathbf{C}(\underline{p})$ is the data dependent correlation matrix

$$\mathbf{C}(\underline{p}) = \begin{bmatrix} \mathbb{E}_P[\eta_{k-1}^2 \mid p_k = p] & \mathbb{E}_P[\eta_{k-1}\eta_{k-2} \mid p_k = p] & \dots & \mathbb{E}_P[\eta_{k-1}\eta_{k-L} \mid p_k = p] \\ \mathbb{E}_P[\eta_{k-2}\eta_{k-1} \mid p_k = p] & \mathbb{E}_P[\eta_{k-2}^2 \mid p_k = p] & \dots & \mathbb{E}_P[\eta_{k-2}\eta_{k-L} \mid p_k = p] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}_P[\eta_{k-L}\eta_{k-1} \mid p_k = p] & \mathbb{E}_P[\eta_{k-L}\eta_{k-2} \mid p_k = p] & \dots & \mathbb{E}_P[\eta_{k-L}^2 \mid p_k = p] \end{bmatrix} \tag{5.2.22}$$

and $\underline{v}(\underline{p})$ is the data dependent vector

$$\underline{v}(\underline{p}) = \begin{bmatrix} \mathbb{E}_P[\eta_k\eta_{k-1} \mid p_k = p] \\ \mathbb{E}_P[\eta_k\eta_{k-2} \mid p_k = p] \\ \vdots \\ \mathbb{E}_P[\eta_k\eta_{k-L} \mid p_k = p] \end{bmatrix} \tag{5.2.23}$$

Finally set the partial derivative w.r.t. $\sigma(\underline{p})$ to zero

$$\begin{aligned} 0 &= 2^D \frac{\partial}{\partial \sigma(\underline{p})} \mathbb{E}_P[BM_k(\underline{n})] \\ &= \frac{1}{\sigma(\underline{p})} - \frac{1}{\sigma^3(\underline{p})} \mathbb{E}_P \left[\left(n_k - \sum_{l=1}^{L} b_l(\underline{p}) n_{k-l} - \mu(\underline{p}) \right)^2 \mid p_k = p \right] \end{aligned} \tag{5.2.24}$$

Hence

$$\begin{aligned}
\sigma(\underline{p})^2 &= \mathbb{E}_P\left[\left(n_k - \sum_{l=1}^{L} b_l(\underline{p}) n_{k-l} - \mu(\underline{p})\right)^2 \mid p_k = p\right] \\
&= \mathbb{E}_P\left[\left(n_k - \sum_{l=1}^{L} b_l(\underline{p}) n_{k-l} - \mathbb{E}_P\left[n_k \mid p_k = p\right]\right.\right. \\
&\quad \left.\left. - \sum_{l=1}^{L} b_l(\underline{p}) \mathbb{E}_P\left[n_{k-l} \mid p_k = p\right]\right)^2 \mid p_k = p\right] \\
&= \mathbb{E}_P\left[\left(\eta_k - \sum_{l=1}^{L} b_l(\underline{p}) \eta_{k-l}\right)^2 \mid p_k = p\right] \\
&= \mathbb{E}_P\left[\eta_k^2 \mid p_k = p\right] - 2\sum_{l=1}^{L} b_l(\underline{p}) \mathbb{E}_P\left[\eta_k \eta_{k-l} \mid p_k = p\right] \\
&\quad + \sum_{l=1}^{L}\sum_{j=1}^{L} b_l(\underline{p}) b_j(\underline{p}) \mathbb{E}_P\left[\eta_{k-l}\eta_{k-j} \mid p_k = p\right] \\
&= \mathbb{E}_P\left[\eta_k^2 \mid p_k = p\right] - 2\underline{b}(\underline{p})^T \underline{v}(\underline{p}) + \underline{b}(\underline{p})^T \mathbf{C}(\underline{p}) \underline{b}(\underline{p})
\end{aligned} \tag{5.2.25}$$

Then using (5.2.21), we can eliminate $\underline{b}(\underline{p})$ to give

$$\begin{aligned}
\sigma(\underline{p})^2 &= \mathbb{E}_P\left[\eta_k^2 \mid p_k = p\right] - 2(\mathbf{C}(\underline{p})^{-1}\underline{v}(\underline{p}))^T \underline{v}(\underline{p}) + (\mathbf{C}(\underline{p})^{-1}\underline{v}(\underline{p}))^T \mathbf{C}(\underline{p}) \mathbf{C}(\underline{p})^{-1} \underline{v}(\underline{p}) \\
&= \mathbb{E}_P\left[\eta_k^2 \mid p_k = p\right] - 2\underline{v}(\underline{p})^T (\mathbf{C}(\underline{p})^{-1})^T \underline{v}(\underline{p}) + \underline{v}(\underline{p})^T (\mathbf{C}(\underline{p})^{-1})^T \underline{v}(\underline{p}) \\
&= \mathbb{E}_P\left[\eta_k^2 \mid p_k = p\right] - \underline{v}(\underline{p})^T (\mathbf{C}(\underline{p})^{-1})^T \underline{v}(\underline{p}) \\
&= \mathbb{E}_P\left[\eta_k^2 \mid p_k = p\right] - \underline{v}(\underline{p})^T \mathbf{C}(\underline{p})^{-1} \underline{v}(\underline{p})
\end{aligned} \tag{5.2.26}$$

since the correlation matrix $\mathbf{C}(\underline{p})$ is symmetric.

5.2.2 Simulation Results

The following plot shows the relative performance of a data dependent auto-regressive detector compared to a data independent auto-regressive detector and a white noise detector.

The choice of target for the white noise detector was determined using the minimum mean squared error (MMSE) criteria, whilst the target for the auto-regressive detectors was chosen as the best performing target obtained from a target search.

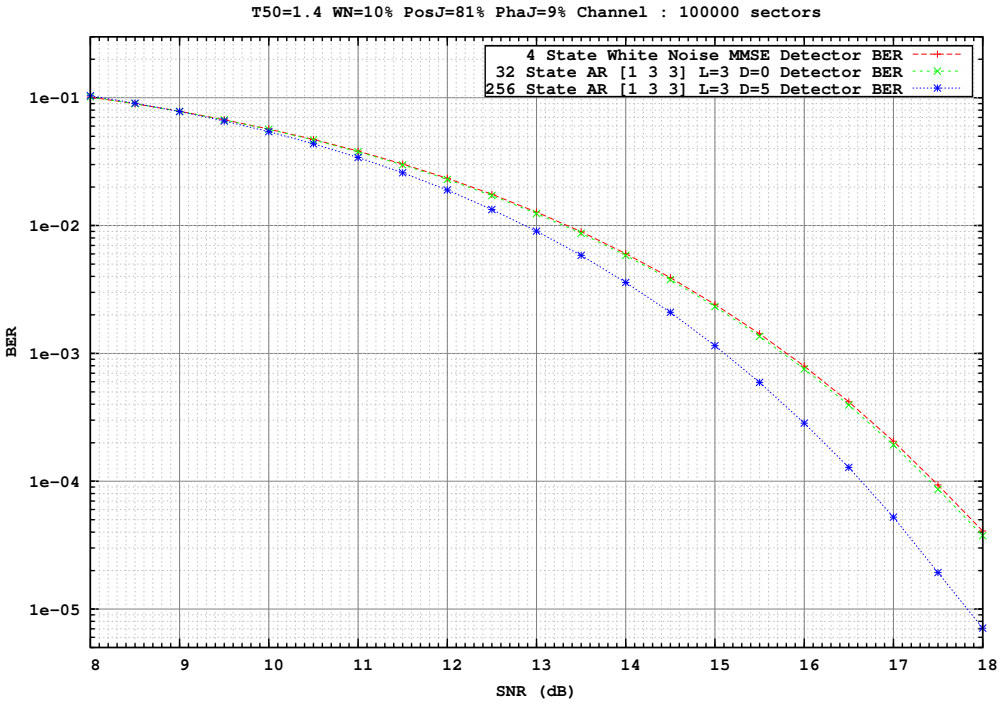


Figure 5.2: Comparison of white noise MMSE detector, data independent auto-regressive detector and data dependent auto-regressive detector.

5.2.3 Limitations

As per the data independent auto-regressive detector, the data dependent auto-regressive detector requires at least 2^{I+L} states.

But in addition, one can see from (5.2.8) that the data patterns used by the branch metric depends on bits x_{i-L-D}^{i}. Therefore the detector must have at least 2^{L+D} states. Hence the data dependent auto-regressive detector requires $2^{L+\max\{I,D\}}$ states.

Again we have the problem that complexity increases exponentially with the correlation length, and pattern length (if sufficiently large). Also note that the correlation coefficients must be stored for each data pattern, and selected between in the implementation, thereby increasing complexity further.

However, the performance gain using data dependent auto-regressive detectors in compelling. Therefore a trade-off must be achieved between complexity and performance.

One trivial suboptimal approach to the problem was proposed in [44]. The idea is to approximate data dependent noise model parameters (L, D) with data dependent noise model with parameters (L', D') such that $L' + D' < L + D$.

Unfortunately, this approach does not always solve the problem of complexity as numerical experiments show a sharp decline in performance as $L+D$ in the model used to describe noise signals falls below a certain threshold value. For perpendicular channel these threshold values turn out to be $L+D = 7$, which means that the corresponding data dependent detector is still impractical.

The following plot shows the relative performance of a data dependent auto-regressive detector with $L = 3$ and $D = 5$ compared to a simplified data dependent auto-regressive detector with $L = 2$ and $D = 2$.

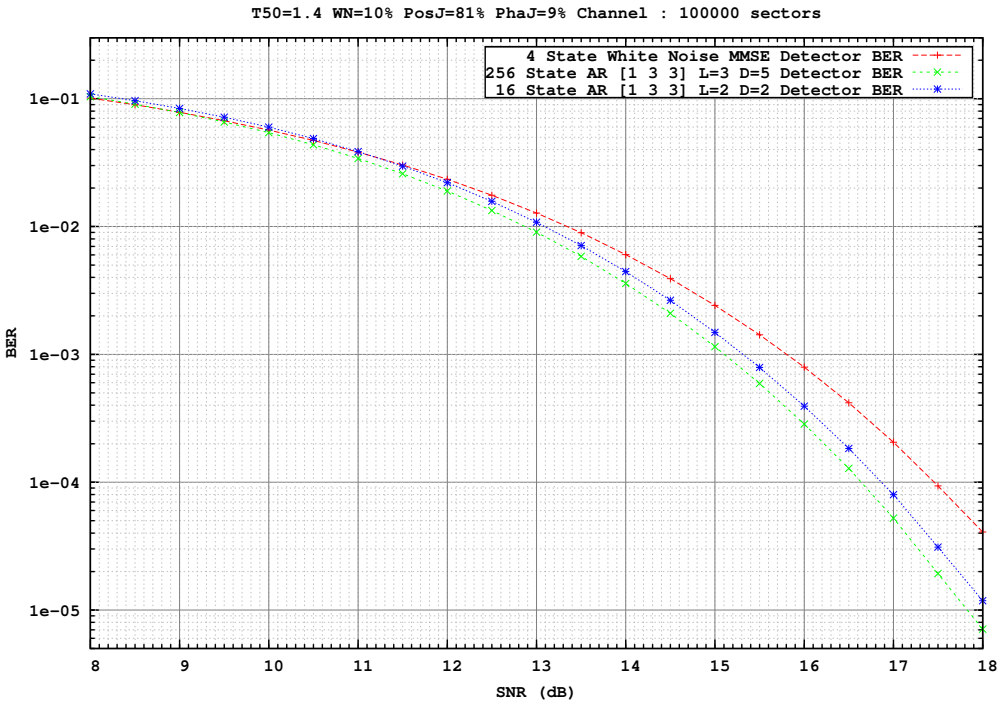


Figure 5.3: Comparison of data dependent auto-regressive detectors with different parameters.

5.3 Block Diagonal Detectors

In order to avoid increasing the number of states, we must ensure the branch metric does not depend on noise estimates from before the current block. The simplest way of achieving this is to neglect inter-block correlations in a high radix Viterbi detector altogether. Mathematically, this corresponds to approximating banded variance matrix of noise signal with a block-diagonal

one. If detector's radix is sufficiently high, the relative contribution of inter-block correlations is small compared with in-block correlations. As a result, the performance of block diagonal detectors approach optimal in the limit of large block sizes. Unfortunately, the rate of convergence is very slow and even radix-64 block-diagonal detector was outperformed significantly by optimal detectors in our numerical simulations.

This approach was originally considered by Altekar and Wolf [43].

The simplest question to ask is: what is the best approximation of the noise statistics with block diagonal statistics? To avoid cluttered notations we will present the derivation of the answer for a particular case of data-independent stationary correlated Gaussian noise and just state the generalization to data dependent case.

Let $\mathbf{C}^{-1}$ be the infinite Toeplitz variance matrix of linear correlated noise. Let $\mathbf{C}_B^{-1}$ be the $B \times B$ variance matrix of block diagonal approximation of this variance matrix, where B is the block size. Here $\mathbf{C}$ is the correlation matrix of noisy signal, $\mathbf{C}_B$ - $B \times B$ correlation matrix corresponding to a block of a block-diagonal approximation. The problem is to find $\mathbf{C}_B^{-1}$ given $\mathbf{C}^{-1}$.

Consider the string of noise samples $\underline{n} = n_i{}_{i=-\infty}^{i=\infty}$, where in order to avoid ill-defined expressions we assume that only finitely many n_i's are non-zero. The probability density of this string is

$$P(\underline{n}) = \frac{1}{Z} e^{-\frac{1}{2}\langle \underline{n}, \mathbf{C}^{-1}\underline{n}\rangle}, \tag{5.3.1}$$

where Z is a normalization constant.

The probability density of the same string in block-diagonal approximation is

$$P_B(\underline{n}) = \prod_{i=-\infty}^{\infty} \sqrt{\frac{1}{(2\pi)^n \det(V_B)}} e^{-\frac{1}{2}\langle n_{i\cdot B}^{(i+1)\cdot B-1}, \mathbf{C}_B^{-1} n_{i\cdot B}^{(i+1)\cdot B-1}\rangle} \tag{5.3.2}$$

The block diagonal distribution P_B which is the closest to P in information-theoretic sense is the one for which the relative entropy

$$D_{KL}[P \mid P_B](V, V_B) \equiv \int \prod_i dn_i P(\underline{n}) \ln\left(\frac{P(\underline{n})}{P_B(\underline{n})}\right) \tag{5.3.3}$$

is minimal. For such a distribution,

$$0 = \frac{\partial}{\partial V_B} D_{KL}[P \mid P_B](V, V_B) = -\int \prod_i dn_i \frac{P}{P_B} \frac{\partial P_B}{\partial V_B} \tag{5.3.4}$$

Using (5.3.2) one can re-write the extremum condition as follows:

$$\langle n_i n_j \rangle_P = \frac{\partial}{\partial V_{B\ ij}} \ln(\det(\mathbf{C}_B^{-1})),\ 1 \le i, j \le B. \tag{5.3.5}$$

Here $\langle \ldots \rangle_P$ denotes averaging with respect to probability density P. As a consequence of Krammer's rule, the r. h. s. of (5.3.5) is just $\overline{V}_B^{-1} \equiv \mathbf{C}_B$. The l. h. s. of (5.3.5) is the exact $B \times B$ correlation matrix of distribution P. We therefore arrive at an intuitively pleasing result that block-diagonal distribution which is the closest to a given distribution P is characterized by

the variance matrix

$$\overline{\underline{V}}_B = \mathbf{C}_B^{-1}, \tag{5.3.6}$$

where $\mathbf{C}_B$ is the exact $B \times B$ correlation matrix of the distribution we are trying to approximate. Note that Gaussianity of P didn't play any role in our derivation.

The generalization of (5.3.6) to data dependent case is straightforward: variance matrix conditioned on a particular data sequence within a block is

$$\overline{\underline{V}}_B \mid_{x_1,x_2,\ldots x_B} = \Big(E(n_i n_j \mid x_1, x_2, \ldots, x_B) \Big)^{-1}, \tag{5.3.7}$$

where the expression in the r. h. s. of (5.3.7) is correlation function of the distribution we are approximating conditioned on the data-sequence $x_1, x_2, \ldots, x_B$.

The expression for the branch metric of block-diagonal maximum likelihood detector is obtained by taking the natural logarithm of distribution (5.3.2):

$$-2 \ln P_B(\underline{n}) = \sum_i BM_i(n_{iB}, n_{iB+1}, \ldots n_{(i+1)B-1}), \tag{5.3.8}$$

where

$$BM_i(n_{iB}, n_{iB+1}, \ldots n_{(i+1)B-1}) = \langle n_{i\cdot B}^{(i+1)\cdot B-1}, \mathbf{C}_B^{-1} n_{i\cdot B}^{(i+1)\cdot B-1} \rangle \tag{5.3.9}$$

is the weight of the path consisting of B branches belonging to the i'th block.

5.3.1 Simulation Results

The following plot shows the relative performance of a data independent block diagonal detector compared to a white noise detector.

The choice of target for the white noise detector was determined using the minimum mean squared error (MMSE) criteria, whilst the target for the block diagonal detectors was chosen as the best performing target obtained from a target search.

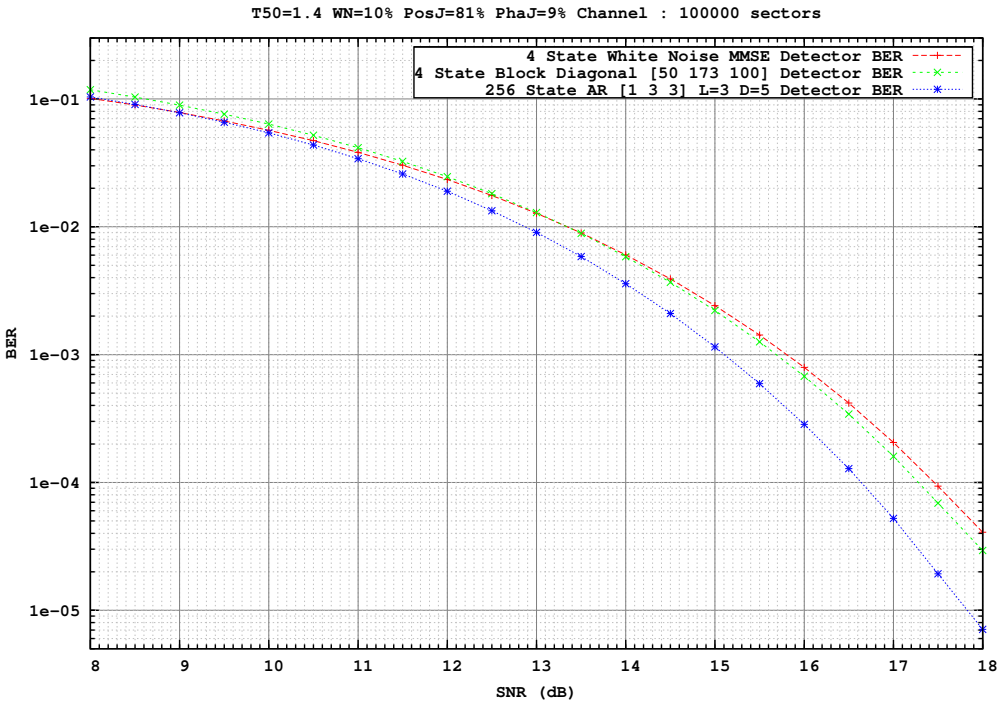


Figure 5.4: Comparison of white noise MMSE detector and block diagonal detector and data dependent auto-regressive detector.

5.3.2 Limitations

Block diagonal detectors only require the same 2^I states as white noise detectors, however the loss of performance due to neglecting the inter-block correlations make such detectors a poor compromise.

An improved block diagonal detector with noise prediction is described in [6].

5.4 Data Dependent Noise Predictive Detector

In the auto-regressive detectors described above, any data bit referenced in the branch metric can be determined exactly from the state and transition.

Noise predictive detectors aim to reduce the number of states by estimating previous data bits rather than determining them exactly from the state information.

This can be achieved by performing a *local traceback* along the surviving path to the current state. The local traceback is performed in exactly the same way as the global traceback, using the survivor decisions stored from previous time steps.

In particular fix the number of states to 2^I, where $I < L + D$ is the ISI length. Then each branch has only the $I + 1$ latest data bits available, but the branch metric requires knowledge of the latest $L + D + 1$ bits. Therefore the previous $L + D - I$ bits must be obtained by local traceback.

Note that local traceback is sub-optimal since we only consider the path specified by the local traceback, and ignore the other $2^{L+D-I}-1$ contenders.

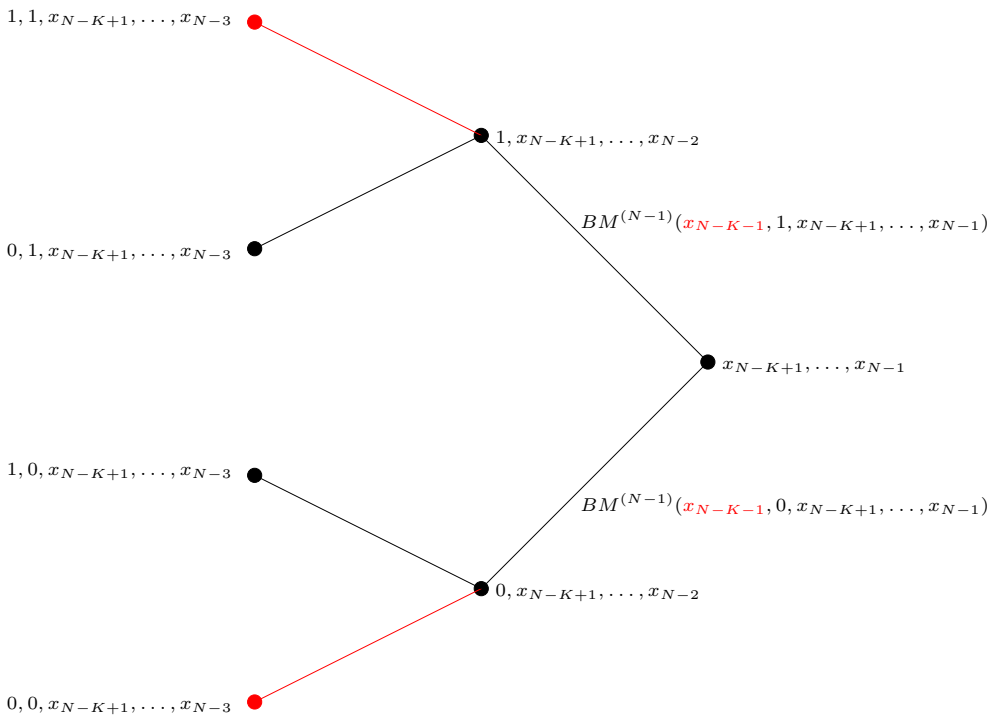


Figure 5.5: Local traceback of single data bit x_{N-K-1}.

5.4.1 Simulation Results

The following plot shows the relative performance of a data dependent noise predictive detector compared to a data dependent auto-regressive detector and a white noise detector.

The choice of target for the white noise detector was determined using the minimum mean squared error (MMSE) criteria, whilst the target for the data dependent detectors was chosen as the best performing target obtained from a target search.

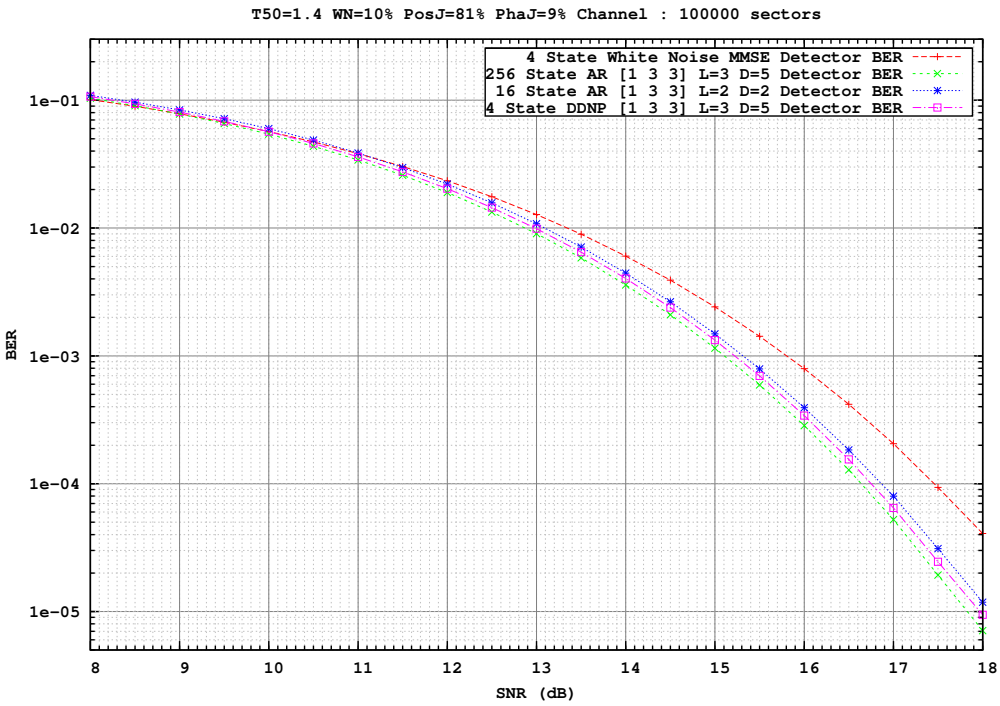


Figure 5.6: Comparison of white noise MMSE detector, data dependent auto-regressive detector and data dependent noise predictive detector.

5.4.2 Limitations

DDNP detectors only require the same 2^I states as white noise detectors, and therefore solve the problem of exponentially increasing complexity with increasing correlation length.

Unfortunately, the local traceback cannot occur until the previous decisions at each state have been made, since the pattern depends on this decision.

This makes implementation impractically slow as this dependence results in a long feedback loop. Feedback loops preclude the possibility of pipelining, therefore any logic contained within the feedback loop must be implemented

in a single cycle. The increased complexity of the feedback loop therefore reduces throughput.

5.4.3 Reduced State ISI Predictive Detectors

The idea of using local traceback to compensate for insufficiently large number of states in the trellis, can be extended beyond predicting states for noise prediction to include state information for ISI prediction using the same technique.

In the following example, a 4-state white noise MMSE detector is outperformed by an 8-state white noise MMSE detector. However, the 4-state white noise detector with a single step local traceback almost matches the performance of the standard detector with twice the number of states.

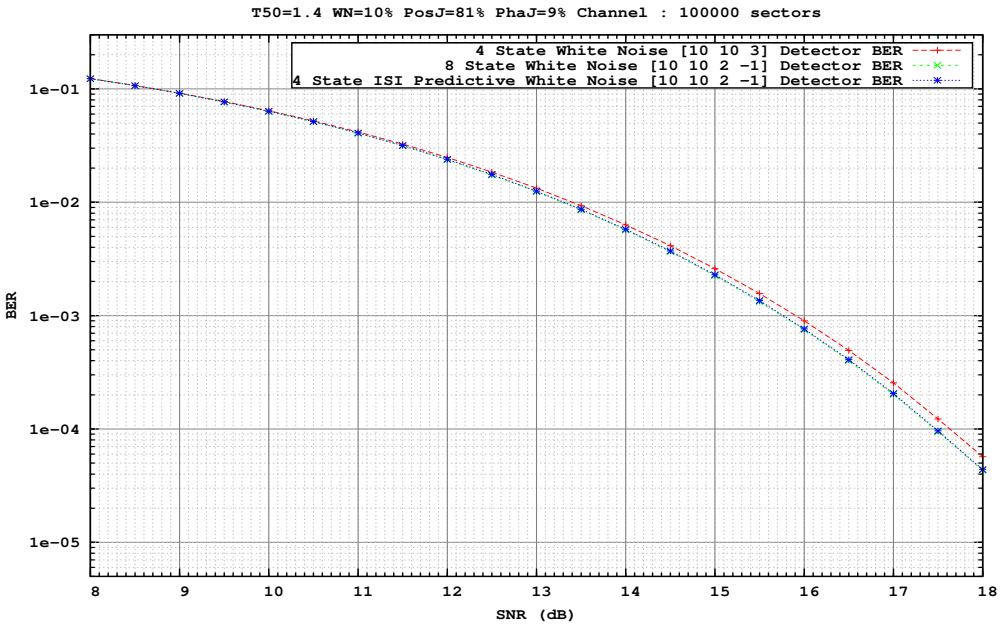


Figure 5.7: Comparison of 4 and 8 state white noise MMSE detectors against a 4 state ISI predictive white noise detector.

5.5 Double Detectors

We have seen that data dependent noise predictive detectors with the same number of states as a white noise detector, perform comparably to data dependent auto-regressive detectors with 2^{L+D-I} times more states.

However, data dependent noise predictive detectors contain a long feedback path between the add compare select unit and the branch metric unit.

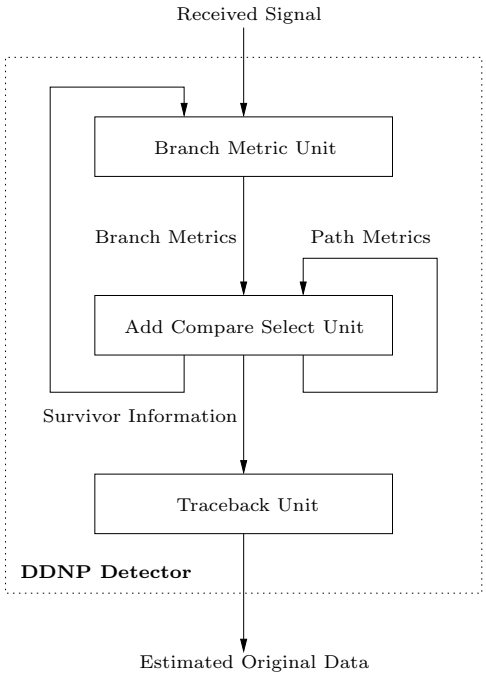


Figure 5.8: DDNP detector showing long noise predictive loop between ACS and BMU.

Therefore to make their implementation practical, the feedback path between the add-compare-select and the local traceback must be eliminated.

The idea of the double detector is to provide an estimate of the missing data bits without performing local traceback by using a pre-detector before the

main detector [1, 4].

The *pre−detector* performs preliminary sequence detection using a simplified branch metric, and feeds these decisions to the main detector, which can then perform a more accurate detection using the more advanced branch metric.

For example, suppose that we would like to compute the branch metric of the branch $x_{i-4}x_{i-3}x_{i-2}x_{i-1}x_i$ of the main detector. Assuming that $I = 4$ and $L = 4$, we also need to know bits $x_{i-8}x_{i-7}x_{i-6}x_{i-5}$ to perform this computation. These can be obtained by a 4-step local trace-back on the trellis of the two-state pre-detector starting from state x_{i-4} at time $i - 4$.

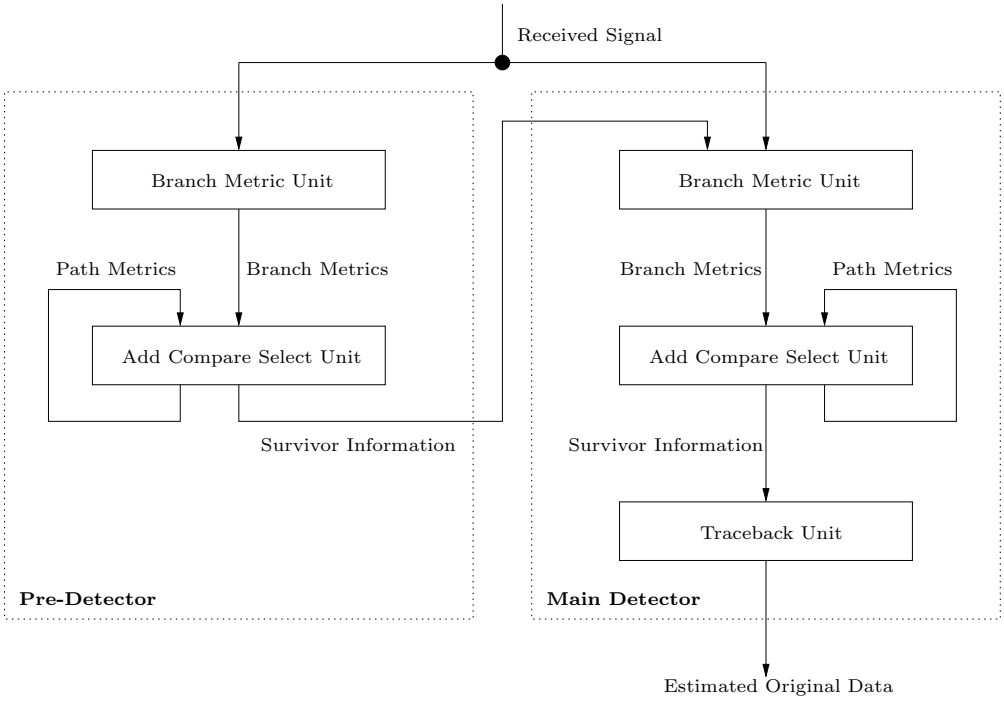


Figure 5.9: Double detector replaces noise predictive loop with survivor information from pre-detector.

Simulation results show that the double detector perform close to DDNP

detectors, which in turn perform close to the optimal data dependent auto-regressive detectors, but without the complexity limitation of the auto-regressive detector, or the implementation issues due to the long feedback path of the DDNP detector.

Note that we can employ more than just a single pre-detector. Multiple detectors can be connected with the survivor information feeding the branch metric unit on the next detector, thus providing increasingly accurate survivor information for the final main detector.

5.5.1 Simulation Results

The following plot shows the relative performance of a double detector detector compared to a data dependent auto-regressive detector, a data dependent noise predictive detector and a white noise detector.

The choice of target for the white noise detector was determined using the minimum mean squared error (MMSE) criteria, whilst the target for the data dependent detectors was chosen as the best performing target obtained from a target search.

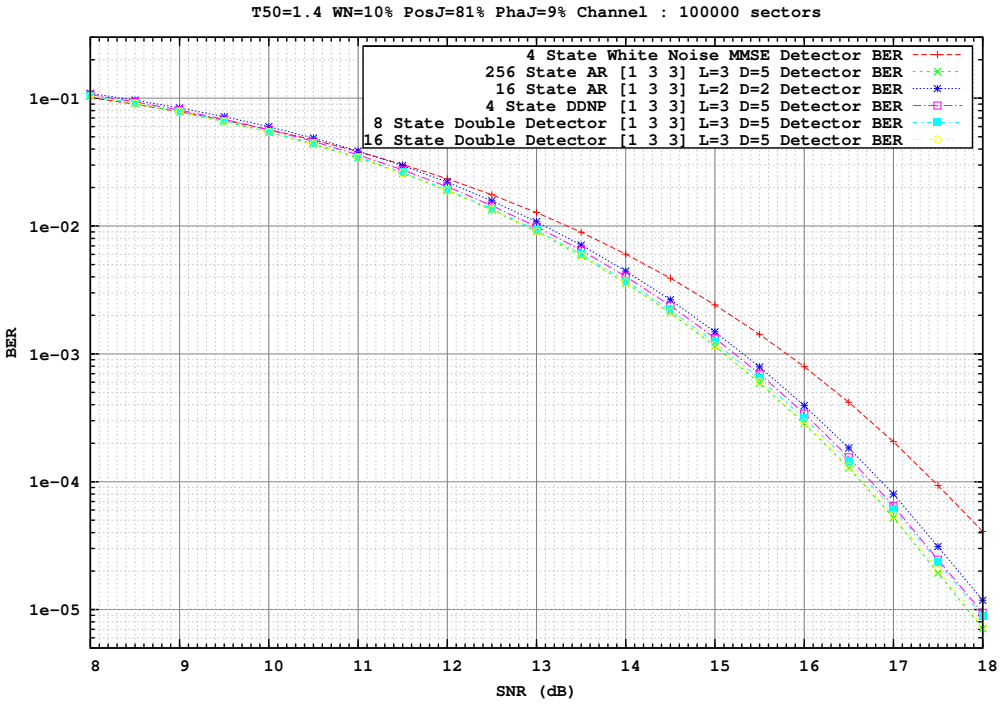


Figure 5.10: Comparison of white noise MMSE detector, data dependent auto-regressive detector, data dependent noise predictive detector and double detector.

Chapter 6

Cost Function

In this chapter we introduce a method of accurately estimating the bit error rate of a maximum likelihood detector with specified parameters (such as equaliser coefficients and ISI target), from only the statistics of the received signal.

Such an estimate is important, as it provides a theoretic means of determining the optimal parameters for the detector, without the need for numerical simulation.

In this chapter we shall only consider the problem of choosing the optimal ISI target for a white noise detector. However this approach can be extended to more complex detectors which require computation of other parameters, such as the noise predictive finite impulse response filter coefficients in auto-regressive detectors.

6.1 Choosing ISI target and equaliser coefficients

Ideally, we aim to choose ISI target and equaliser coefficients which minimise the BER. However this can only be achieved by numerical simulation over a large search space. Therefore we use alternative criteria based on statistics of the noise, which allow us to choose the parameters without the need for numerical simulation.

The first minimisation criteria we shall consider is minimising the power of the noise relative to the power of the signal, i.e. maximising the signal to noise ratio (SNR).

However, we shall find that a second criteria which minimises the noise strength without regard to the power of the signal, but instead subject to a monic constraint, significantly outperforms maximising SNR. This criteria is referred to as minimum mean squared error criteria (MMSE) [47].

We shall investigate why MMSE outperforms maximising SNR, and show that MMSE is equivalent to whitening the noise.

Finally we shall introduce an estimate for BER from the statistics of the noise, and use this criteria to find the optimal parameters, and show that this approach outperforms MMSE.

6.1.1 Maximum SNR Criteria

Before we decode the received signal using the Viterbi algorithm, we equalise the signal to a specific ISI target using a finite impulse response filter. We therefore need to choose an ISI target and equaliser coefficients which give maximum performance after the Viterbi.

Suppose the ISI target has K coefficients, where K is the constraint length, the equaliser has $2T+1$ taps and the original data $x_t \in \{\pm 1\}$. We choose the ISI target coefficients g_k and the equaliser coefficients f_n such that the SNR is maximised, where SNR is defined by

$$SNR = 10\log_{10}\left(\frac{\mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i}\right)^2\right]}{\mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right)^2\right]}\right) \tag{6.1.1}$$

Note that if all coefficients g_i and f_j are scaled by some fixed constant, then the SNR remains unchanged. Therefore maximising the SNR is equivalent to minimising the denominator subject to the constraint

$$\sum_{i=0}^{K-1} g_i^2 = 1 \tag{6.1.2}$$

since

$$\mathbb{E}\left[x_t x_{t-i}\right] = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.1.3}$$

Introducing a Lagrange multiplier λ for the constraint, we need to minimise

the function

$$R(\underline{g}, \underline{f}, \lambda) = \mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right)^2\right] - \lambda\left(\sum_{i=0}^{K-1} g_i^2 - 1\right) \quad (6.1.4)$$

Taking partial derivatives with respect to each ISI target coefficient g_k and filter coefficient f_n

$$\begin{aligned}
\frac{\partial R}{\partial \lambda} &= \sum_{i=0}^{K-1} g_i^2 - 1 \\
\frac{\partial R}{\partial g_k} &= 2\mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right) x_{t-k}\right] - 2\lambda g_k \quad \text{for } k = 0, \ldots, K-1 \\
\frac{\partial R}{\partial f_n} &= -2\mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right) r_{t-n}\right] \qquad \text{for } n = -T, \ldots, T
\end{aligned} \quad (6.1.5)$$

At the minimum, the partial derivatives $\frac{\partial R}{\partial \lambda} = \frac{\partial R}{\partial g_k} = \frac{\partial R}{\partial f_n} = 0$, therefore we must simultaneously solve

$$\begin{aligned}
\sum_{i=0}^{K-1} g_i^2 &= 1 \\
\sum_{i=0}^{K-1} g_i \mathbb{E}\left[x_{t-i} x_{t-k}\right] - \lambda g_k &= \sum_{j=-T}^{T} f_j \mathbb{E}\left[r_{t-j} x_{t-k}\right] \quad \text{for } k = 0, \ldots, K-1 \\
\sum_{i=0}^{K-1} g_i \mathbb{E}\left[x_{t-i} r_{t-n}\right] &= \sum_{j=-T}^{T} f_j \mathbb{E}\left[r_{t-j} r_{t-n}\right] \qquad \text{for } n = -T, \ldots, T
\end{aligned} \quad (6.1.6)$$

which we can rewrite as

$$\begin{aligned} \sum_{i=0}^{K-1} g_i^2 &= 1 \\ \sum_{i=0}^{K-1} X_{k,i} g_i - \lambda g_k &= \sum_{j=-T}^{T} V_{j,k} f_j \quad \text{for } k = 0, \ldots, K-1 \\ \sum_{i=0}^{K-1} V_{n,i} g_i &= \sum_{j=-T}^{T} C_{n,j} f_j \qquad \text{for } n = -T, \ldots, T \end{aligned} \tag{6.1.7}$$

where

$$\begin{aligned} X_{k,i} &= \mathbb{E}\left[x_{t-i} x_{t-k}\right] \\ V_{n,i} &= \mathbb{E}\left[x_{t-i} r_{t-n}\right] \\ C_{n,j} &= \mathbb{E}\left[r_{t-j} r_{t-n}\right] \end{aligned} \tag{6.1.8}$$

Moreover, we can express (6.1.7) in matrix form as

$$\begin{aligned} \underline{g}^T \underline{g} &= 1 \\ (\mathbf{X} - \lambda \mathbf{I}) \underline{g} &= \mathbf{V}^T \underline{f} \\ \mathbf{V} \underline{g} &= \mathbf{C} \underline{f} \end{aligned} \tag{6.1.9}$$

where

$$\begin{aligned} \mathbf{X} &: \{0, \ldots, K-1\} \times \{0, \ldots, K-1\} \to \mathbb{R} \\ \mathbf{V} &: \{-T, \ldots, T\} \times \{0, \ldots, K-1\} \to \mathbb{R} \\ \mathbf{C} &: \{-T, \ldots, T\} \times \{-T, \ldots, T\} \to \mathbb{R} \\ \mathbf{I} &: \{0, \ldots, K-1\} \times \{0, \ldots, K-1\} \to \mathbb{R} \end{aligned} \tag{6.1.10}$$

from which we can eliminate $\underline{f} = \mathbf{C}^{-1} \mathbf{V} \underline{g}$ giving

$$(\mathbf{X} - \mathbf{V}^T \mathbf{C}^{-1} \mathbf{V} - \lambda \mathbf{I}) \underline{g} = \underline{0} \tag{6.1.11}$$

Solutions to (6.1.11) are therefore given by the eigenvectors of matrix $\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V}$. We can determine the value of the noise function (6.1.4) at a solution $\underline{g}$ to be

$$\begin{aligned} R &= \underline{g}^T\mathbf{X}\underline{g} - 2\underline{f}^T\mathbf{V}\underline{g} + \underline{f}^T\mathbf{C}\underline{f} \\ &= \underline{g}^T\mathbf{X}\underline{g} - 2(\mathbf{C}^{-1}\mathbf{V}\underline{g})^T\mathbf{V}\underline{g} + (\mathbf{C}^{-1}\mathbf{V}\underline{g})^T\mathbf{C}(\mathbf{C}^{-1}\mathbf{V}\underline{g}) \\ &= \underline{g}^T\mathbf{X}\underline{g} - 2\underline{g}^T\mathbf{V}^T{\mathbf{C}^{-1}}^T\mathbf{V}\underline{g} + \underline{g}^T\mathbf{V}^T{\mathbf{C}^{-1}}^T\mathbf{C}\mathbf{C}^{-1}\mathbf{V}\underline{g} \\ &= \underline{g}^T\mathbf{X}\underline{g} - \underline{g}^T\mathbf{V}^T{\mathbf{C}^{-1}}^T\mathbf{V}\underline{g} \\ &= \underline{g}^T(\mathbf{X} - \mathbf{V}^T{\mathbf{C}^{-1}}^T\mathbf{V})\underline{g} \qquad (6.1.12) \\ &= \underline{g}^T(\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V})\underline{g} \qquad \text{since } C^T = C \\ &= \underline{g}^T(\lambda\mathbf{I})\underline{g} \\ &= \lambda\underline{g}^T\underline{g} \\ &= \lambda \end{aligned}$$

which is the eigenvalue corresponding to the eigenvector $\underline{g}$. Hence the global minimum is given by the eigenvector corresponding to the smallest eigenvalue of the matrix $\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V}$.

6.1.2 Minimum Mean Squared Error Criteria

Let us choose as the minimisation criteria, to minimise the noise energy

$$E = \mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right)^2\right] \qquad (6.1.13)$$

subject to the constraint $g_0 = 1$. Therefore we need to minimise the function

$$R(\underline{g}, \underline{f}) = \mathbb{E}\left[\left(x_t + \sum_{i=1}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right)^2\right] \tag{6.1.14}$$

Taking partial derivatives with respect to each ISI target coefficient g_k and filter coefficient f_n

$$\begin{aligned}
\frac{\partial R}{\partial g_k} &= 2\mathbb{E}\left[\left(x_t + \sum_{i=1}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right) x_{t-k}\right] \quad \text{for } k = 1, \ldots, K-1 \\
\frac{\partial R}{\partial f_n} &= -2\mathbb{E}\left[\left(x_t + \sum_{i=1}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right) r_{t-n}\right] \quad \text{for } n = -T, \ldots, T
\end{aligned} \tag{6.1.15}$$

At the minimum, the partial derivatives $\frac{\partial R}{\partial g_k} = \frac{\partial R}{\partial f_n} = 0$, therefore we must simultaneously solve

$$\begin{aligned}
\mathbb{E}\left[x_t x_{t-k}\right] + \sum_{i=1}^{K-1} g_i \mathbb{E}\left[x_{t-i} x_{t-k}\right] &= \sum_{j=-T}^{T} f_j \mathbb{E}\left[r_{t-j} x_{t-k}\right] \quad \text{for } k = 1, \ldots, K-1 \\
\mathbb{E}\left[x_t r_{t-n}\right] + \sum_{i=1}^{K-1} g_i \mathbb{E}\left[x_{t-i} r_{t-n}\right] &= \sum_{j=-T}^{T} f_j \mathbb{E}\left[r_{t-j} r_{t-n}\right] \quad \text{for } n = -T, \ldots, T
\end{aligned} \tag{6.1.16}$$

which we can rewrite as

$$\begin{aligned}
x_k + \sum_{i=1}^{K-1} X_{k,i} g_i &= \sum_{j=-T}^{T} V_{j,k} f_j \quad \text{for } k = 1, \ldots, K-1 \\
v_n + \sum_{i=1}^{K-1} V_{n,i} g_i &= \sum_{j=-T}^{T} C_{n,j} f_j \quad \text{for } n = -T, \ldots, T
\end{aligned} \tag{6.1.17}$$

where

$$\begin{aligned} x_k &= \mathbb{E}\left[x_t x_{t-k}\right] \\ v_n &= \mathbb{E}\left[x_t r_{t-n}\right] \\ X_{k,i} &= \mathbb{E}\left[x_{t-i} x_{t-k}\right] \\ V_{n,i} &= \mathbb{E}\left[x_{t-i} r_{t-n}\right] \\ C_{n,j} &= \mathbb{E}\left[r_{t-j} r_{t-n}\right] \end{aligned} \tag{6.1.18}$$

Moreover, we can express (6.1.17) in matrix form as

$$\begin{aligned} \underline{x} + \mathbf{X}\underline{g} &= \mathbf{V}^T \underline{f} \\ \underline{v} + \mathbf{V}\underline{g} &= \mathbf{C}\underline{f} \end{aligned} \tag{6.1.19}$$

where

$$\begin{aligned} \underline{x} &: \{1, \ldots, K-1\} \to \mathbb{R} \\ \underline{v} &: \{-T, \ldots, T\} \to \mathbb{R} \\ \mathbf{X} &: \{1, \ldots, K-1\} \times \{1, \ldots, K-1\} \to \mathbb{R} \\ \mathbf{V} &: \{-T, \ldots, T\} \times \{1, \ldots, K-1\} \to \mathbb{R} \\ \mathbf{C} &: \{-T, \ldots, T\} \times \{-T, \ldots, T\} \to \mathbb{R} \end{aligned} \tag{6.1.20}$$

from which we can eliminate $\underline{f} = \mathbf{C}^{-1}(\underline{v} + \mathbf{V}\underline{g})$ giving

$$\begin{aligned} \underline{x} + \mathbf{X}\underline{g} &= \mathbf{V}^T\mathbf{C}^{-1}(\underline{v} + \mathbf{V}\underline{g}) \\ (\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V})\underline{g} &= \mathbf{V}^T\mathbf{C}^{-1}\underline{v} - \underline{x} \\ \underline{g} &= (\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V})^{-1}(\mathbf{V}^T\mathbf{C}^{-1}\underline{v} - \underline{x}) \end{aligned} \tag{6.1.21}$$

Therefore we can determine the equaliser coefficients and ISI target as follows

$$\begin{aligned} \underline{f} &= \mathbf{C}^{-1}(\underline{v} + \mathbf{V}(\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V})^{-1}(\mathbf{V}^T\mathbf{C}^{-1}\underline{v} - \underline{x})) \\ \underline{g} &= (\mathbf{X} - \mathbf{V}^T\mathbf{C}^{-1}\mathbf{V})^{-1}(\mathbf{V}^T\mathbf{C}^{-1}\underline{v} - \underline{x}) \end{aligned} \tag{6.1.22}$$

6.1.3 Noise Spectra

The following plots show for various noise models, the noise spectrum of the unequalised received signal and the noise equalised to targets selected by maximum SNR and MMSE criterion.

Before considering a realistic channel, consider the ideal [1 3 3 1] ISI channel with only additive white Gaussian noise.

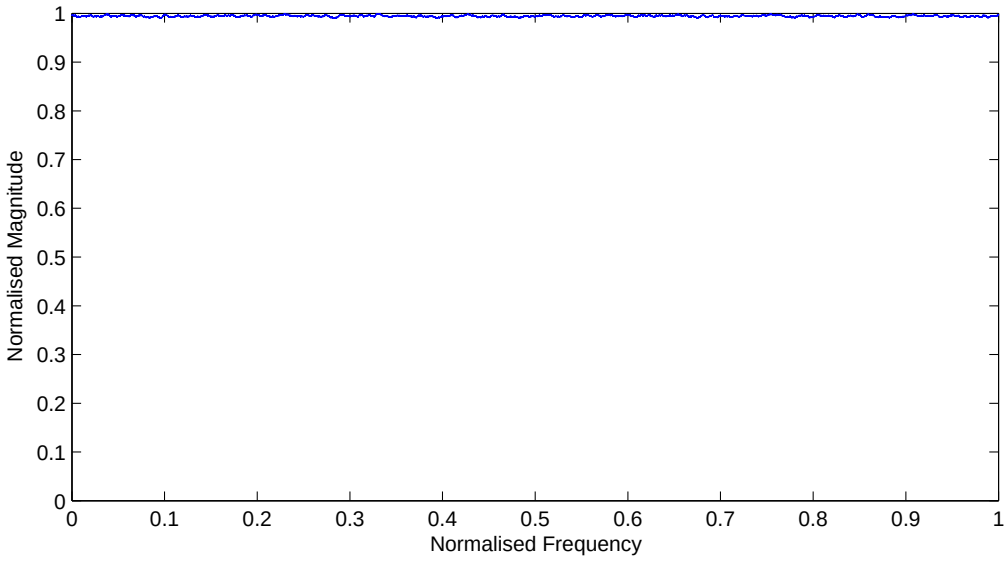


Figure 6.1: Spectrum of unequalised received signal for [1 3 3 1] AWGN channel.

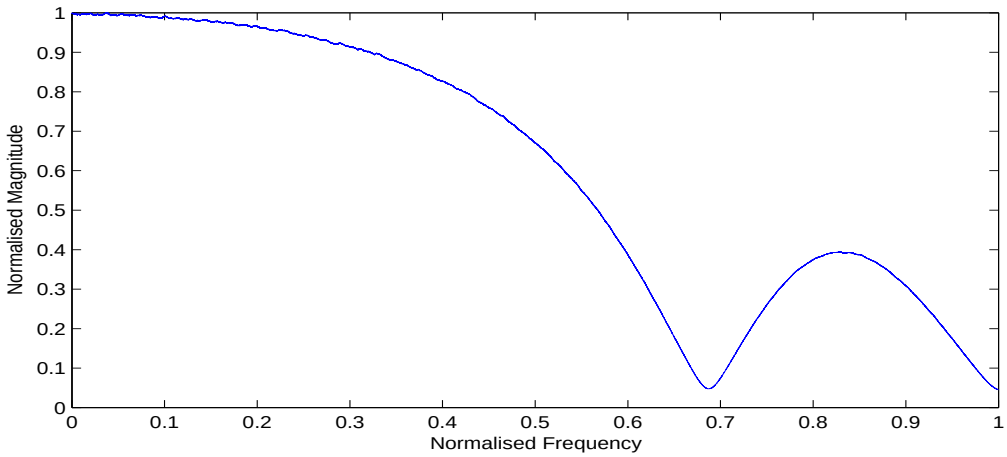


Figure 6.2: Spectrum of received signal equalised to target [1 2.1 2.1 1] as chosen by SNR criteria for [1 3 3 1] AWGN channel.

Note that the spectrum of the received signal equalised to the target as chosen by the minimum SNR criteria shows the noise is not white, but strongly correlated. A white noise detector is only optimal for white noise, therefore the above spectrum may explain the poor performance of this target (see table 6.1).

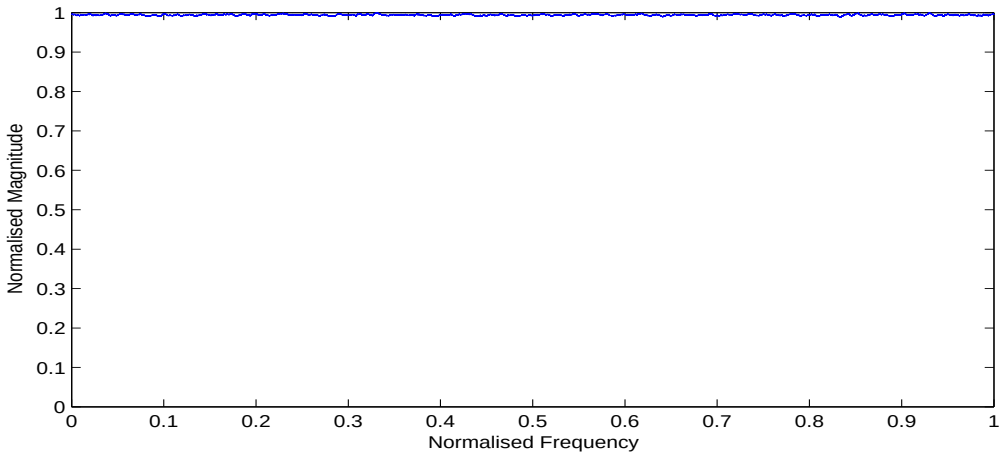


Figure 6.3: Spectrum of received signal equalised to target [1 1.1 0.6 0.1] as chosen by MMSE criteria for [1 3 3 1] AWGN channel.

The following table shows that the MMSE criteria chooses a target which performs almost as well as the ideal target for this trivial channel, but the coefficients of the target differ significantly. Also the maximum SNR criteria demonstrably increases the SNR after the equaliser, but this actually yields poor performance.

Criteria	Target	Experimental SNR	BER
Ideal	[1 3 3 1]	12.5 dB	2.84×10^{-3}
SNR	[1 2.1 2.1 1]	15.3 dB	6.98×10^{-3}
MMSE	[1 1.1 0.6 0.1]	12.7 dB	2.89×10^{-3}

Table 6.1: Comparison of performance at 12.5 dB for [1 3 3 1] AWGN channel.

Next consider a realistic channel as described in (4.1.1) with 70% media noise, where the media noise consists of 100% position jitter, and the error function describes the isolated transitions.

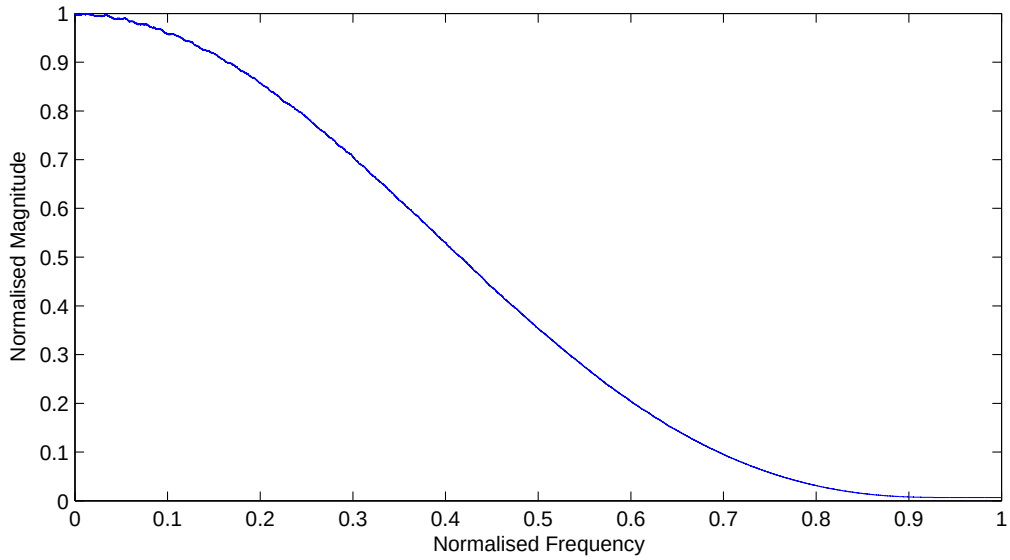


Figure 6.4: Spectrum of unequalised received signal for 70% position jitter erf channel.

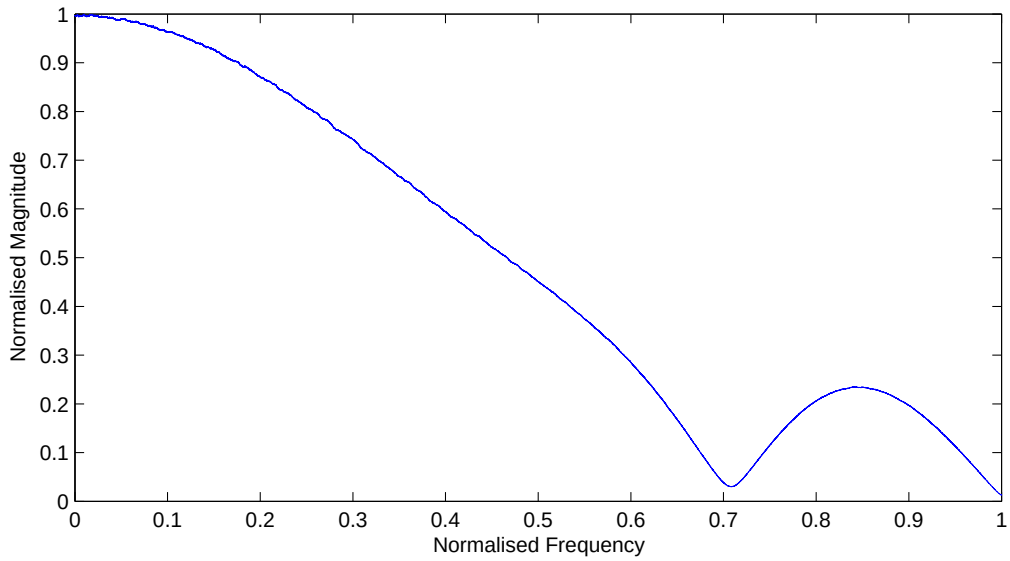


Figure 6.5: Spectrum of received signal equalised to target [1.00 2.21 2.21 1.00] as chosen by SNR criteria for 70% position jitter erf channel.

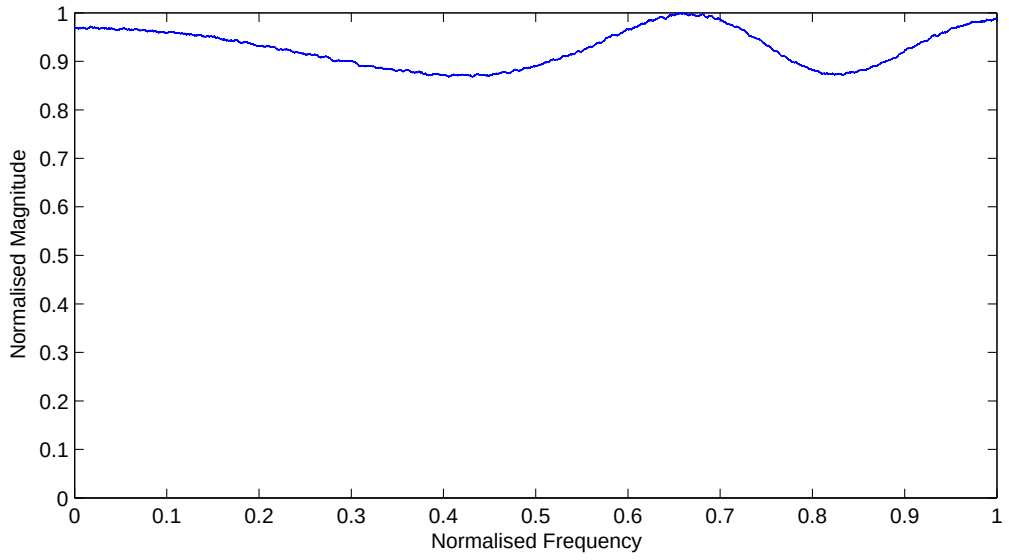


Figure 6.6: Spectrum of received signal equalised to target [1.00 0.97 0.32 -0.04] as chosen by MMSE criteria for 70% position jitter erf channel.

Again for the realistic channel, the maximum SNR criteria achieves the high-

est SNR, but MMSE achieves better performance.

Criteria	Target	Experimental SNR	BER
SNR	[1.00 2.21 2.21 1.00]	14.5 dB	2.23×10^{-2}
MMSE	[1.00 0.97 0.32 -0.04]	11.8 dB	6.98×10^{-3}

Table 6.2: Comparison of performance at 14.0 dB for 70% position jitter erf channel.

Next consider a jitter dominated channel as described in (4.1.1) with 90% media noise, where the media noise consists of 90% position jitter and 10% phase jitter, and the error function describes the isolated transitions.

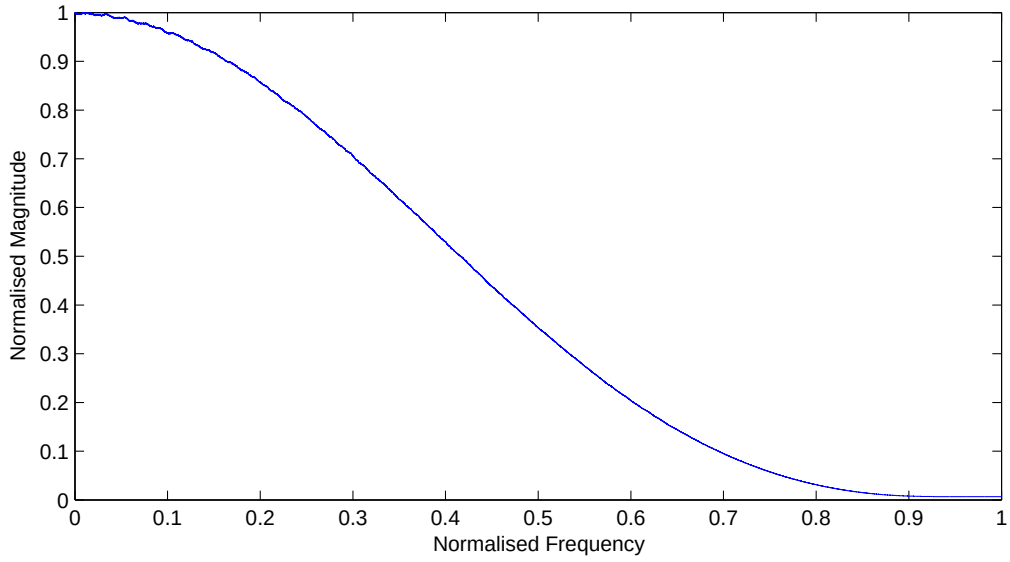


Figure 6.7: Spectrum of unequalised received signal for 90% media noise 90% position jitter erf channel.

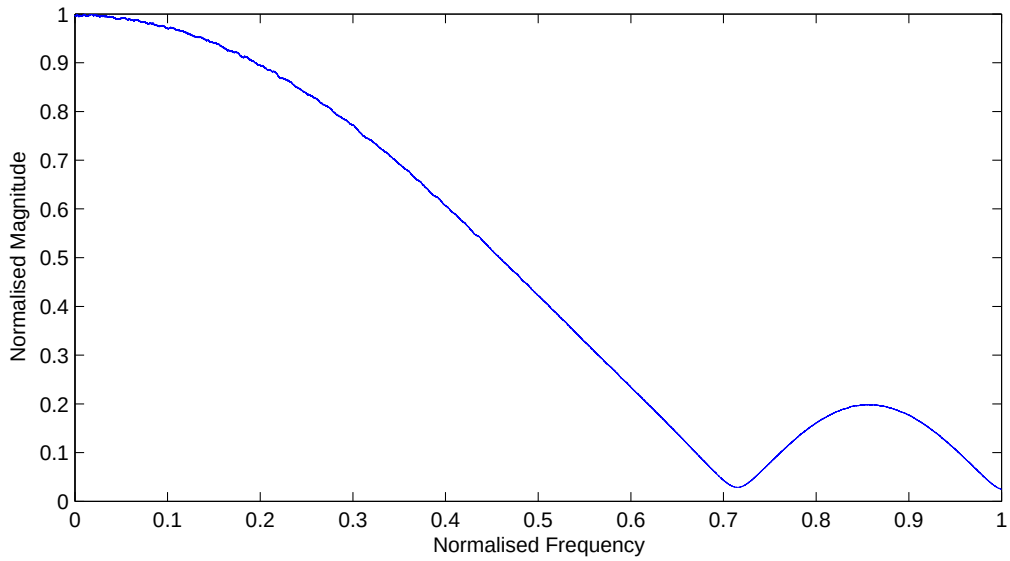


Figure 6.8: Spectrum of received signal equalised to target [1.00 2.23 2.22 0.98] as chosen by SNR criteria for 90% media noise 90% position jitter erf channel.

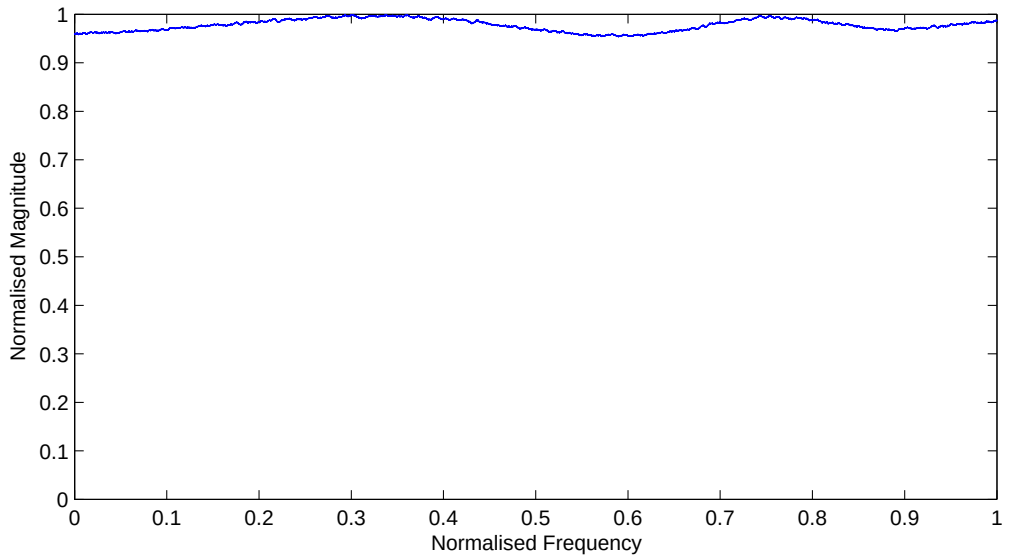


Figure 6.9: Spectrum of received signal equalised to target [1.00 0.87 0.15 -0.07] as chosen by MMSE criteria for 90% media noise 90% position jitter erf channel.

The following plot shows the relative performance of the targets chosen by the maximum SNR and MMSE criteria.

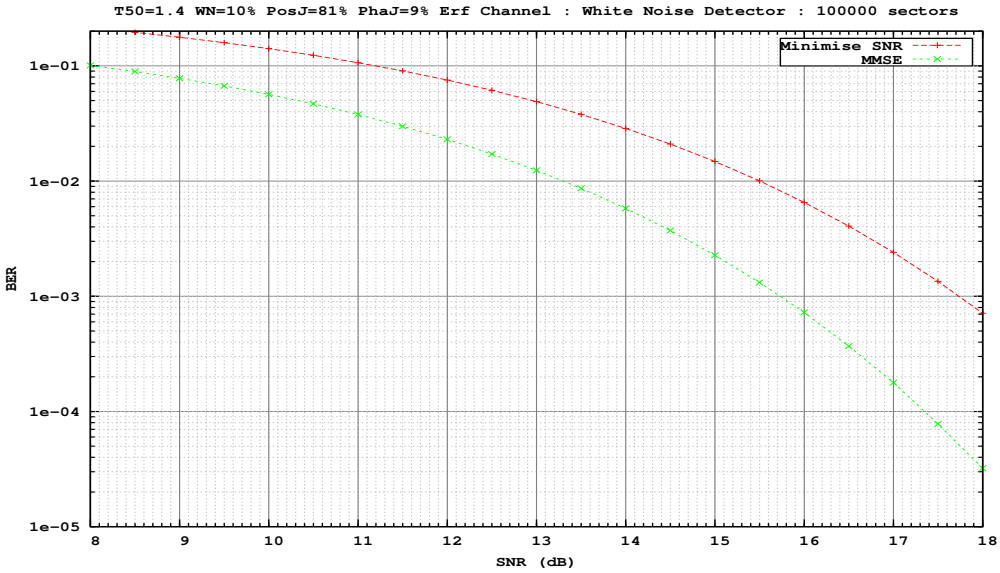


Figure 6.10: Comparison of performance for 90% media noise 90% position jitter erf channel.

Finally consider a channel with the same parameters as above, but which uses hyperbolic tangent to describe the isolated transitions.

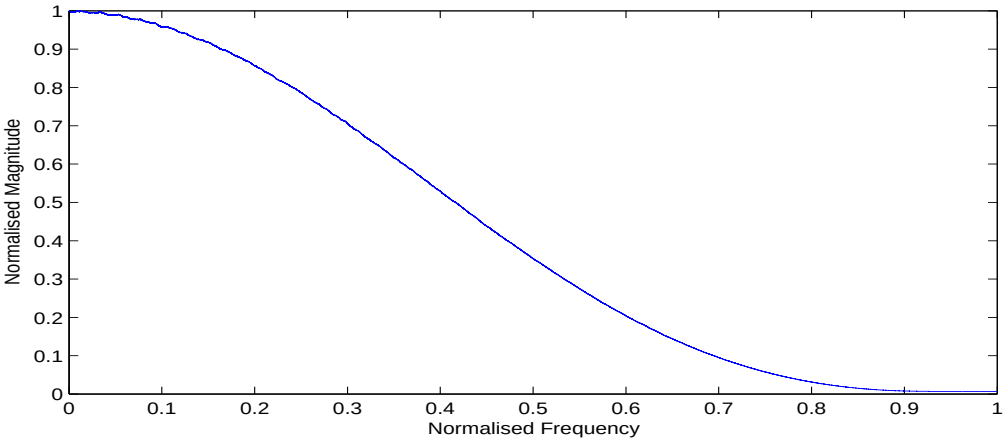


Figure 6.11: Spectrum of unequalised received signal for 90% media noise 90% position jitter tanh channel.

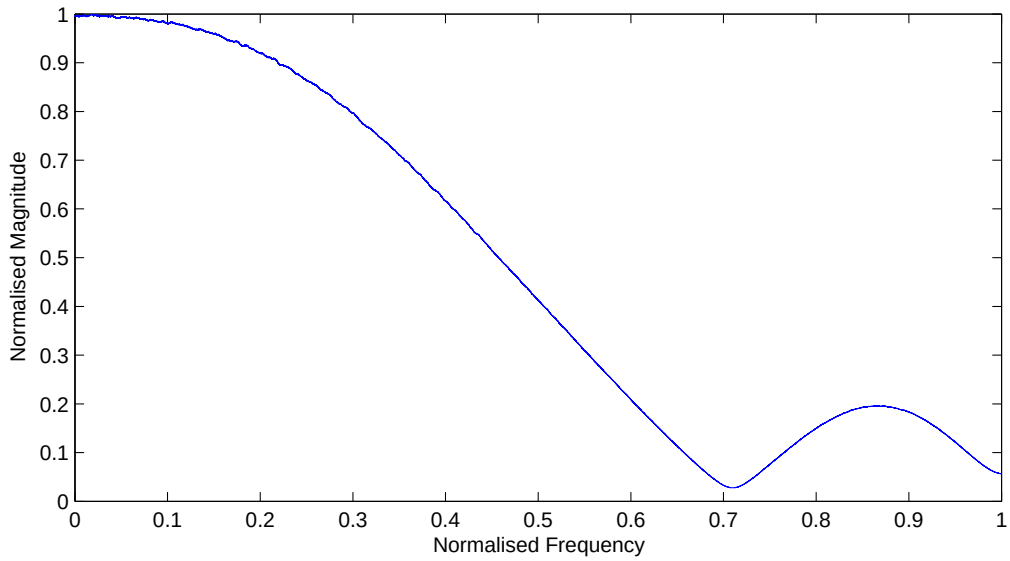


Figure 6.12: Spectrum of received signal equalised to target [1.00 2.19 2.18 0.98] as chosen by SNR criteria for 90% media noise 90% position jitter tanh channel.

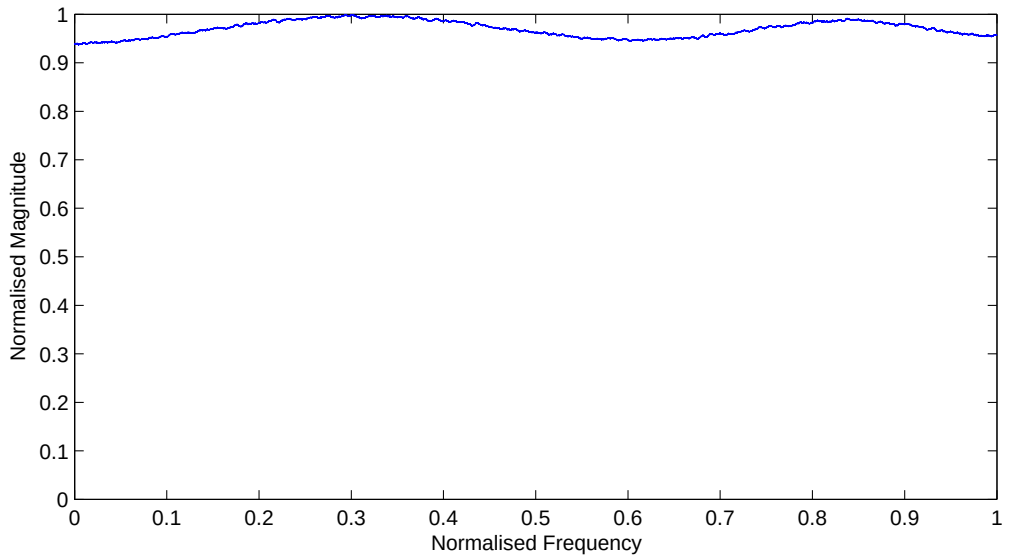


Figure 6.13: Spectrum of received signal equalised to target [1.00 0.81 0.09 -0.03] as chosen by MMSE criteria for 90% media noise 90% position jitter tanh channel.

Criteria	Target	Experimental SNR	BER
SNR	[1.00 2.19 2.18 0.98]	14.8 dB	1.99×10^{-2}
MMSE	[1.00 0.81 0.09 -0.03]	12.3 dB	1.68×10^{-3}

Table 6.3: Comparison of performance at 14.0 dB for 90% media noise 90% position jitter tanh channel.

Note that the spectra for the targets chosen by MMSE are all flat, indicating a whitening of the signal.

6.1.4 MMSE Equivalence to Whitening Noise

Suppose we have an equaliser and target of infinite length and we choose the filter and target coefficients to minimise the noise energy, $\mathbb{E}\left[n_t^2\right]$, subject to the constraint $g_0 = 1$ [47], where

$$n_t = \sum_{i=0}^{\infty} g_i x_{t-i} - \sum_{j=-\infty}^{\infty} f_j r_{t-j} \tag{6.1.23}$$

Since we have chosen the filter and target coefficients to minimise the noise energy, we have

$$\begin{aligned} \frac{\partial}{\partial g_k}\mathbb{E}\left[n_t^2\right] &= 0 \quad \text{for } k > 0 \\ \frac{\partial}{\partial f_n}\mathbb{E}\left[n_t^2\right] &= 0 \quad \text{for all } n \end{aligned} \tag{6.1.24}$$

Let us now consider the correlations between the noise n_t and the coefficients g_k, f_n by differentiating $\mathbb{E}\left[n_t^2\right]$.

$$\begin{aligned} 0 = \frac{1}{2}\frac{\partial}{\partial g_k}\mathbb{E}\left[n_t^2\right] = \mathbb{E}\left[n_t \frac{\partial n_t}{\partial g_k}\right] &= \mathbb{E}\left[n_t x_{t-k}\right] \quad \text{for } k > 0 \\ 0 = -\frac{1}{2}\frac{\partial}{\partial f_n}\mathbb{E}\left[n_t^2\right] = -\mathbb{E}\left[n_t \frac{\partial n_t}{\partial f_n}\right] &= \mathbb{E}\left[n_t r_{t-n}\right] \quad \text{for all } n \end{aligned} \tag{6.1.25}$$

Therefore if we consider the correlations between noise samples n_t and n_{t-k} for $k > 0$ then

$$\begin{aligned}\mathbb{E}\left[n_t n_{t-k}\right] &= \mathbb{E}\left[n_t\left(\sum_{i=0}^{\infty} g_i x_{t-k-i} - \sum_{j=-\infty}^{\infty} f_j r_{t-k-j}\right)\right] \\ &= \sum_{i=0}^{\infty} g_i \mathbb{E}\left[n_t x_{t-k-i}\right] - \sum_{j=-\infty}^{\infty} f_j \mathbb{E}\left[n_t r_{t-k-j}\right] \\ &= 0\end{aligned} \tag{6.1.26}$$

Hence minimising the noise energy is equivalent to whitening the noise. This can be confirmed by observing the spectrum of the noise. Note how using the MMSE criteria results in a flat spectrum 6.3, 6.6, 6.9, 6.13, in comparison with the maximum SNR criteria 6.2, 6.5, 6.8, 6.12.

6.1.5 Minimise Bit Error Rate Criteria

In this section, we aim to create an accurate estimate of the BER based only on the statistics of the received signal and the coefficients of the target polynomial. We can then use this estimate as part of a steepest descent algorithm to find the target coefficients which minimise BER.

The BER is defined as

$$BER = \frac{1}{N}\sum_{i=0}^{N-1} \mathbb{P}(\hat{x}_i \neq x_i) \tag{6.1.27}$$

where x_i are the original data, and $\hat{x}_i$ are the estimates provided by the Viterbi detector. Assuming translational invariance of the channel, $\mathbb{P}(\hat{x}_i \neq$

$x_i) = \mathbb{P}(\hat{x}_j \neq x_j)$, therefore we can restrict ourselves to considerations at a specific time i.

$$BER = \mathbb{P}(\hat{x}_i \neq x_i) \tag{6.1.28}$$

Fix the original and received signals by conditioning over all possible original sequences, $\underline{x}$, and received signals, $\underline{r}$.

$$BER = \sum_{\underline{x},\underline{r}} \mathbb{P}(\hat{x}_i \neq x_i \mid \underline{x}, \underline{r}) \mathbb{P}(\underline{x}, \underline{r}) \tag{6.1.29}$$

The estimates $\hat{\underline{x}}$ provided by the Viterbi detector correspond to the path taken by the shortest path through the trellis. Therefore an error occurs at time i if and only if the i-th bit of the shortest path differs from the original data.

$$\begin{aligned} \mathbb{P}(\hat{x}_i \neq x_i \mid \underline{x}, \underline{r}) &= \sum_{\left\{P \in \mathcal{P} \mid P^{(i)} \neq x_i\right\}} \mathbb{P}\left(PM(P) = \min_{P_j \in \mathcal{P}} \{PM(P_j)\} \mid \underline{x}, \underline{r}\right) \\ &= \sum_{\left\{P \in \mathcal{P} \mid P^{(i)} \neq x_i\right\}} \mathbb{P}(PM(P) \leq PM(P_j), \forall P_j \in \mathcal{P} \mid \underline{x}, \underline{r}) \end{aligned} \tag{6.1.30}$$

where $\mathcal{P}$ is the set of all paths and $P^{(i)}$ refers to the value of the i-th bit of the path P. Separating the event when P is the correct path through the trellis $\underline{x}$ we find that

$$\begin{aligned} \mathbb{P}(\hat{x}_i \neq x_i \mid \underline{x}, \underline{r}) = \sum_{\left\{P \in \mathcal{P} \mid P^{(i)} \neq x_i\right\}} & \mathbb{P}(PM(P) \leq PM(\underline{x}) \mid \underline{x}, \underline{r}) \times \\ & \mathbb{P}(PM(P) \leq PM(P_j), \forall P_j \neq \underline{x} \mid PM(P) \leq PM(\underline{x}), \underline{x}, \underline{r}) \end{aligned} \tag{6.1.31}$$

Note that the event $\{PM(P) \leq PM(P_j), \forall P_j \neq \underline{x} \mid PM(P) \leq PM(\underline{x}), \underline{x}, \underline{r}\}$ does

not occur if and only if there exists some j such that

$$PM(P_j) < PM(P) \leq PM(\underline{x}) \tag{6.1.32}$$

This implies there exists at least two paths through the trellis with path metric lower than the path metric of the correct path. As the SNR increases, the probability of multiple paths having a greater likelihood than the correct path decreases. In particular in the absence of noise, the probability of paths having path metric less than that of the correct path is zero, consequently

$$\lim_{SNR\to\infty} \mathbb{P}(PM(P) \leq PM(P_j), \forall P_j \neq \underline{x} \mid PM(P) \leq PM(\underline{x}), \underline{x}, \underline{r}) = 1 \tag{6.1.33}$$

Therefore we can use the following bound for BER, which is a tight bound for high SNR.

$$\mathbb{P}(\hat{x}_i \neq x_i \mid \underline{x}, \underline{r}) \leq \sum_{\left\{P\in\mathcal{P} \mid P^{(i)} \neq x_i\right\}} \mathbb{P}(PM(P) \leq PM(\underline{x}) \mid \underline{x}, \underline{r}) \tag{6.1.34}$$

Since we have assumed the channel is translationally invariant, instead of considering every path $P \in \mathcal{P}$ that differs from the correct path at time i, we can consider all paths which re-converge with the correct path $\underline{x}$ at a specified time, say t.

In the original formulation, a divergent path that re-converges at time t and contains multiple errors, will be offset to each position where an error occurs, and each such offset will contribute separately to the summation. In our reformulation, we count each path exactly once, and weight that path by

the number of positions where $P^{(j)} \neq x_j$. Therefore

$$\mathbb{P}(\hat{x}_i \neq x_i \mid \underline{x}, \underline{r}) \leq \sum_{\substack{\left\{P \in \mathcal{P} \mid P^{(j)} = x_j \text{ for } j > t\right\} \\ P \neq \underline{x}}} W(P)\mathbb{P}(PM(P) \leq PM(\underline{x}) \mid \underline{x}, \underline{r}) \tag{6.1.35}$$

where $W(P) = \left|\left\{j \mid P^{(j)} \neq x_j\right\}\right|$ is the weight of the path.

For numerical computation, we require a finite sum approximation to the above infinite sum. We choose to limit the number of paths considered to those which diverge for less than a fixed number of steps. Therefore we have the following approximation

$$\begin{aligned} &\mathbb{P}(\hat{x}_i \neq x_i \mid \underline{x}, \underline{r}) \\ &\approx \sum_{\substack{\left\{P \in \mathcal{P} \mid P^{(j)} = x_j \text{ for } j < t-L \text{ and } j > t\right\} \\ P \neq \underline{x}}} W(P)\mathbb{P}(PM(P) \leq PM(\underline{x}) \mid \underline{x}, \underline{r}) \end{aligned} \tag{6.1.36}$$

This will be a good approximation since the nature of the Viterbi algorithm is for paths to re-converge with high probability after relatively few time steps.

Let us now fix the original data and received signal, and consider for each path P the probability $\mathbb{P}(PM(P) \leq PM(\underline{x}))$. Let $PM_{ideal} = PM(\underline{x})$ be the path metric corresponding to the ideal path through the trellis, and let PM_{error} be the path metric corresponding to the path through the trellis caused by an error event, which re-converges with the ideal path at time t.

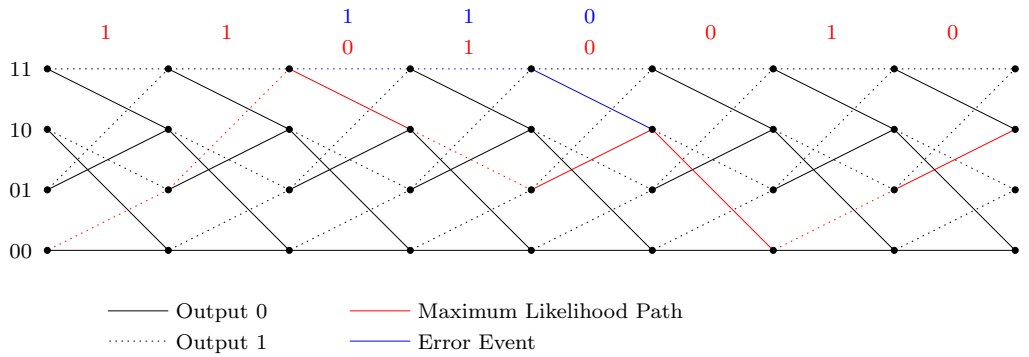


Figure 6.14: 4 state trellis showing maximum likelihood path and error event.

Note that the length of the divergence between the ideal and the error path is $K + L - 1$, where K is the constraint length and L is the length of the error event.

The error path will be chosen instead of the correct path if

$$PM_{error} < PM_{actual} \tag{6.1.37}$$

We need to estimate the probability

$$\mathbb{P}(PM_{error} < PM_{actual}) = \mathbb{P}(PM_{error} - PM_{actual} < 0) \tag{6.1.38}$$

Therefore we must consider the path metric difference

$$PM_{error} - PM_{actual} = \sum_{i=-\infty}^{\infty} (y_i - W_i)^2 - \sum_{i=-\infty}^{\infty} (y_i - I_i)^2 \tag{6.1.39}$$

where y_i are the equalised received signals, I_i are the ideal branch labels and

W_i are the branch labels corresponding to the error event path. Then

$$\begin{aligned} PM_{error} - PM_{actual} &= \sum_{i=-\infty}^{\infty} W_i^2 - I_i^2 - 2(W_i - I_i)y_i \\ &= \sum_{i=t-K-L+2}^{t} W_i^2 - I_i^2 - 2(W_i - I_i)y_i \\ &= \sum_{i=0}^{K-L+2} W_{t-i}^2 - I_{t-i}^2 - 2(W_{t-i} - I_{t-i})y_{t-i} \\ &= \sum_{i=0}^{K-L+2} \beta_i - 2\alpha_i y_{t-i} \end{aligned} \tag{6.1.40}$$

since $W_i = I_i$ for $i > t$ and $i < t - K - L + 2$ (recalling the length of the divergence is $K + L - 1$), where $\alpha_i = W_{t-i} - I_{t-i}$ and $\beta_i = W_{t-i}^2 - I_{t-i}^2 = (W_{t-i} - I_{t-i})(W_{t-i} + I_{t-i})$.

Note that α_i and β_i depend only on the original data and the target coefficients, therefore if we make the pattern sufficiently long, we may consider these values as constants for a particular pattern. Since W_i depends on K bits of the original data, we require a total pattern length of $2K + L - 2$ to cover the whole error event.

Example 6.1.1 (1-bit error event)**.** Let us consider the example of a 1-bit error event ($L = 1$) on a 4-state detector ($K = 3$). The path corresponding to the error event diverges from the correct path when the error occurs, and re-converges 3 steps later, say at time t at state x_{t-1}, x_t.

The paths therefore diverged from state x_{t-4}, x_{t-3} with the correct path passing through states x_{t-3}, x_{t-2} and x_{t-2}, x_{t-1}, whilst the error path passes through states $x_{t-3}, \overline{x_{t-2}}$ and $\overline{x_{t-2}}, x_{t-1}$.

Therefore we can calculate the branch labels corresponding the the actual and error paths (note that we are assuming $x_i \in \{\pm 1\}$, therefore $\overline{x_i} = -x_i$)

$$\begin{aligned} I_t &= g_0 x_t + g_1 x_{t-1} + g_2 x_{t-2} & W_t &= g_0 x_t + g_1 x_{t-1} - g_2 x_{t-2} \\ I_{t-1} &= g_0 x_{t-1} + g_1 x_{t-2} + g_2 x_{t-3} & W_{t-1} &= g_0 x_{t-1} - g_1 x_{t-2} + g_2 x_{t-3} \\ I_{t-2} &= g_0 x_{t-2} + g_1 x_{t-3} + g_2 x_{t-4} & W_{t-2} &= -g_0 x_{t-2} + g_1 x_{t-3} + g_2 x_{t-4} \end{aligned} \tag{6.1.41}$$

Hence the path metric differences and sums are given by

$$\begin{aligned} W_t - I_t &= -2g_2 x_{t-2} & W_t + I_t &= 2(g_0 x_t + g_1 x_{t-1}) \\ W_{t-1} - I_{t-1} &= -2g_1 x_{t-2} & W_{t-1} + I_{t-1} &= 2(g_0 x_{t-1} + g_2 x_{t-3}) \\ W_{t-2} - I_{t-2} &= -2g_0 x_{t-2} & W_{t-2} + I_{t-2} &= 2(g_1 x_{t-3} + g_2 x_{t-4}) \end{aligned} \tag{6.1.42}$$

From the above, we can compute α_i and β_i for any pattern. For example, consider the alternating pattern $\{x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t\} = \{1, -1, 1, -1, 1\}$

$$\begin{aligned} \alpha_0 &= -2g_2 & \beta_0 &= -4g_2(g_0 + g_1) \\ \alpha_1 &= -2g_1 & \beta_1 &= -4g_1(-g_0 + -g_2) \\ \alpha_2 &= -2g_0 & \beta_2 &= -4g_0(-g_1 + g_2) \end{aligned} \tag{6.1.43}$$

Referring back to (6.1.40), we see that the path metric difference is a linear combination of conditionally Gaussian random variables, therefore the path metric difference is also conditionally Gaussian. This implies that for a given pattern $\underline{x} = x_{t-2K-L+3}, \ldots, x_t$

$$PM_{error} - PM_{actual} \sim N(\mu, \sigma^2) \tag{6.1.44}$$

where the mean and variance are trivially given by

$$\begin{aligned} \mu &= \mathbb{E}\left[PM_{error} - PM_{actual}\right] \\ \sigma^2 &= \mathrm{Var}\left[PM_{error} - PM_{actual}\right] \end{aligned} \tag{6.1.45}$$

Therefore if we subtract the mean and normalise

$$\frac{PM_{error} - PM_{actual} - \mathbb{E}\left[PM_{error} - PM_{actual}\right]}{\sqrt{\mathrm{Var}\left[PM_{error} - PM_{actual}\right]}} \sim N(0,1) \tag{6.1.46}$$

which allows us to express the probability of this particular error event, for a given pattern, as

$$\begin{aligned} &\mathbb{P}(PM_{error} < PM_{actual} \mid \underline{x}) \\ &= \mathbb{P}(PM_{error} - PM_{actual} < 0 \mid \underline{x}) \\ &= \mathbb{P}\left(\frac{PM_{error} - PM_{actual} - \mathbb{E}\left[PM_{error} - PM_{actual}\right]}{\sqrt{\mathrm{Var}\left[PM_{error} - PM_{actual}\right]}} \right. \\ &\quad \left. < -\frac{\mathbb{E}\left[PM_{error} - PM_{actual}\right]}{\sqrt{\mathrm{Var}\left[PM_{error} - PM_{actual}\right]}} \mid \underline{x} \right) \\ &= \frac{1}{2}\,\mathrm{erfc}\left(\frac{\mathbb{E}\left[PM_{error} - PM_{actual}\right]}{\sqrt{\mathrm{Var}\left[PM_{error} - PM_{actual}\right]}} \right) \end{aligned} \tag{6.1.47}$$

We can then average over all patterns to get an estimate of the BER for this particular error event.

$$\mathbb{P}(PM_{error} < PM_{actual}) = \sum_{\underline{x} \in \{\pm 1\}^{2K+L-2}} \mathbb{P}(PM_{error} < PM_{actual} \mid \underline{x}) \tag{6.1.48}$$

In order to compute the above, we need to determine $\mathbb{E}\left[PM_{error} - PM_{actual}\right]$ and $\mathrm{Var}\left[PM_{error} - PM_{actual}\right]$. We will show that these can be determined

directly from the statistics of the noise.

Firstly, let us consider the mean of the path metric difference. We can express this expectation in terms of the statistics of the equalised received signal y_i

$$\begin{aligned}\mathbb{E}\left[PM_{error} - PM_{actual}\right] &= \mathbb{E}\left[\sum_{i=0}^{K-L+2} \beta_i - 2\alpha_i y_{t-i}\right] \\ &= \sum_{i=0}^{K-L+2} \beta_i - 2\alpha_i \mathbb{E}\left[y_{t-i}\right]\end{aligned} \tag{6.1.49}$$

Secondly, let us consider the variance of the path metric difference. This requires the calculation of the square of the path metric difference

$$\begin{aligned}&(PM_{error} - PM_{actual})^2 \\ &= \left(\sum_{i=0}^{K-L+2} \beta_i - 2\alpha_i y_{t-i}\right)^2 \\ &= \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} (\beta_i - 2\alpha_i y_{t-i})(\beta_j - 2\alpha_j y_{t-j}) \\ &= \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \beta_i\beta_j - 2\alpha_i\beta_j y_{t-i} - 2\alpha_j\beta_i y_{t-j} + 4\alpha_i\alpha_j y_{t-i}y_{t-j} \\ &= \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \beta_i\beta_j - 4\alpha_i\beta_j y_{t-i} + 4\alpha_i\alpha_j y_{t-i}y_{t-j}\end{aligned} \tag{6.1.50}$$

Therefore the variance can be determined from the square of the expectation

of the path metric difference

$$\begin{aligned}
&\mathbb{E}^2\left[PM_{error} - PM_{actual}\right] \\
&\quad = \left(\sum_{i=0}^{K-L+2} \beta_i - 2\alpha_i \mathbb{E}\left[y_{t-i}\right]\right)^2 \\
&\quad = \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \beta_i\beta_j - 2\alpha_i\beta_j\mathbb{E}\left[y_{t-i}\right] - 2\alpha_j\beta_i\mathbb{E}\left[y_{t-j}\right] + 4\alpha_i\alpha_j\mathbb{E}\left[y_{t-i}\right]\mathbb{E}\left[y_{t-j}\right] \\
&\quad = \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \beta_i\beta_j - 4\alpha_i\beta_j\mathbb{E}\left[y_{t-i}\right] + 4\alpha_i\alpha_j\mathbb{E}\left[y_{t-i}\right]\mathbb{E}\left[y_{t-j}\right]
\end{aligned} \tag{6.1.51}$$

and the expectation of the squared path metric difference

$$\begin{aligned}
&\mathbb{E}\left[(PM_{error} - PM_{actual})^2\right] \\
&\quad = \mathbb{E}\left[\sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \beta_i\beta_j - 4\alpha_i\beta_j y_{t-i} + 4\alpha_i\alpha_j y_{t-i}y_{t-j}\right] \\
&\quad = \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \beta_i\beta_j - 4\alpha_i\beta_j\mathbb{E}\left[y_{t-i}\right] + 4\alpha_i\alpha_j\mathbb{E}\left[y_{t-i}y_{t-j}\right]
\end{aligned} \tag{6.1.52}$$

which gives the following expression for the variance

$$\begin{aligned}
&\mathrm{Var}\left[PM_{error} - PM_{actual}\right] \\
&\quad = \mathbb{E}\left[(PM_{error} - PM_{actual})^2\right] - \mathbb{E}^2\left[PM_{error} - PM_{actual}\right] \\
&\quad = \sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} 4\alpha_i\alpha_j\mathbb{E}\left[y_{t-i}y_{t-j}\right] - 4\alpha_i\alpha_j\mathbb{E}\left[y_{t-i}\right]\mathbb{E}\left[y_{t-j}\right] \\
&\quad = 4\sum_{i=0}^{K-L+2}\sum_{j=0}^{K-L+2} \alpha_i\alpha_j(\mathbb{E}\left[y_{t-i}y_{t-j}\right] - \mathbb{E}\left[y_{t-i}\right]\mathbb{E}\left[y_{t-j}\right])
\end{aligned} \tag{6.1.53}$$

The above relies on pattern dependent statistics of the equalised received

signal. However, we want to use only the statistics of the actual unequalised received signal. Therefore we must specify the type of equaliser to be used.

A key use for the BER estimate is to provide an analytical means of finding the optimal equaliser and ISI target coefficients. This still leaves a large search space of parameters, however we can reduce the problem to finding only the optimal ISI coefficients if we assume the following data independent equaliser, the coefficients of which can be determined directly from the ISI target and noise statistics (6.1.61).

We choose the filter coefficients f_j to minimise the squared difference between the ideal received signal for a given target, and the filtered received signal, y_t, which is a linear combination of the actual received signal, r_t

$$y_t = \sum_{j=-T}^{T} f_j r_{t-j} \tag{6.1.54}$$

Therefore we need to minimise the function

$$R(\underline{f}) = \mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right)^2\right] \tag{6.1.55}$$

Taking partial derivatives with respect to each filter coefficient f_n

$$\frac{\partial R}{\partial f_n} = -2\mathbb{E}\left[\left(\sum_{i=0}^{K-1} g_i x_{t-i} - \sum_{j=-T}^{T} f_j r_{t-j}\right) r_{t-n}\right] \quad \text{for } n = -T, \ldots, T \tag{6.1.56}$$

At the minimum, the partial derivatives $\frac{\partial R}{\partial f_n} = 0$, therefore we must simulta-

neously solve

$$\sum_{i=0}^{K-1} g_i \mathbb{E}\left[x_{t-i} r_{t-n}\right] = \sum_{j=-T}^{T} f_j \mathbb{E}\left[r_{t-j} r_{t-n}\right] \quad \text{for } n = -T, \ldots, T \tag{6.1.57}$$

which we can rewrite as

$$\sum_{i=0}^{K-1} V_{n,i} g_i = \sum_{j=-T}^{T} C_{n,j} f_j \quad \text{for } n = -T, \ldots, T \tag{6.1.58}$$

where

$$\begin{aligned} V_{n,i} &= \mathbb{E}\left[x_{t-i} r_{t-n}\right] \\ C_{n,j} &= \mathbb{E}\left[r_{t-j} r_{t-n}\right] \end{aligned} \tag{6.1.59}$$

Moreover, we can express (6.1.58) in matrix form as

$$\mathbf{V}\underline{g} = \mathbf{C}\underline{f} \tag{6.1.60}$$

Therefore we can determine the filter coefficients $\underline{f}$ as follows

$$\underline{f} = \mathbf{C}^{-1}\mathbf{V}\underline{g} \tag{6.1.61}$$

Note that $\mathbf{C}^{-1}\mathbf{V}$ is determined only from the original data and the received signal. In particular it is pattern independent and independent of the target.

Let $\mathbf{F}$ be an $(2T+1) \times K$ matrix with entries $F_{i,j}$, such that $\mathbf{F} = \mathbf{C}^{-1}\mathbf{V}$, then we can express the equalised received signal as

$$y_t = \sum_{j=-T}^{T} f_j r_{t-j} = \sum_{j=-T}^{T} \sum_{k=0}^{K-1} F_{j,k} g_k r_{t-j} \tag{6.1.62}$$

Therefore we can rewrite the expressions for the mean and variance of the path metric difference for a given pattern as

$$\begin{aligned}\mathbb{E}\left[PM_{error} - PM_{actual}\right] &= \sum_{i=0}^{K-L+2} \left(\beta_i - 2\alpha_i \mathbb{E}\left[y_{t-i}\right]\right) \\ &= \sum_{i=0}^{K-L+2} \beta_i - 2 \sum_{i=0}^{K-L+2} \sum_{j=-T}^{T} \alpha_i f_j \mathbb{E}\left[r_{t-i-j}\right]\end{aligned} \tag{6.1.63}$$

and

$$\begin{aligned}&\mathrm{Var}\left[PM_{error} - PM_{actual}\right] \\ &= 4 \sum_{i=0}^{K-L+2} \sum_{j=0}^{K-L+2} \alpha_i \alpha_j (\mathbb{E}\left[y_{t-i} y_{t-j}\right] - \mathbb{E}\left[y_{t-i}\right] \mathbb{E}\left[y_{t-j}\right]) \\ &= 4 \sum_{i=0}^{K-L+2} \sum_{j=0}^{K-L+2} \sum_{k=-T}^{T} \sum_{l=-T}^{T} \alpha_i \alpha_j f_k f_l (\mathbb{E}\left[r_{t-i-k} r_{t-j-l}\right] - \mathbb{E}\left[r_{t-i-k}\right] \mathbb{E}\left[r_{t-j-l}\right])\end{aligned} \tag{6.1.64}$$

These expressions may further be simplified to

$$\mathbb{E}\left[PM_{error} - PM_{actual}\right] = \sum_{i=0}^{K-L+2} \beta_i - 2 \sum_{i=-T}^{T+K-L+2} \gamma_i \mathbb{E}\left[r_{t-i}\right] \tag{6.1.65}$$

and

$$\begin{aligned}&\mathrm{Var}\left[PM_{error} - PM_{actual}\right] \\ &= 4 \sum_{i=-T}^{T+K-L+2} \sum_{j=-T}^{T+K-L+2} \gamma_i \gamma_j (\mathbb{E}\left[r_{t-i} r_{t-j}\right] - \mathbb{E}\left[r_{t-i}\right] \mathbb{E}\left[r_{t-j}\right])\end{aligned} \tag{6.1.66}$$

where

$$\gamma_k = \sum_{i+j=k} \alpha_i f_j = \sum_{i+j=k} \sum_{l=0}^{K-1} \alpha_i F_{j,l} g_l \tag{6.1.67}$$

This gives us an expression for the BER of a particular error event in terms of the statistics of the received signal, and the coefficients of the target polynomial.

To demonstrate the accuracy of our estimate of BER, consider the following plot which shows the BER as predicted by the cost function against the actual BER as measured for numerical simulation for the ideal [1 3 3 1] ISI channel, using a white noise detector with target [1 3 3 1].

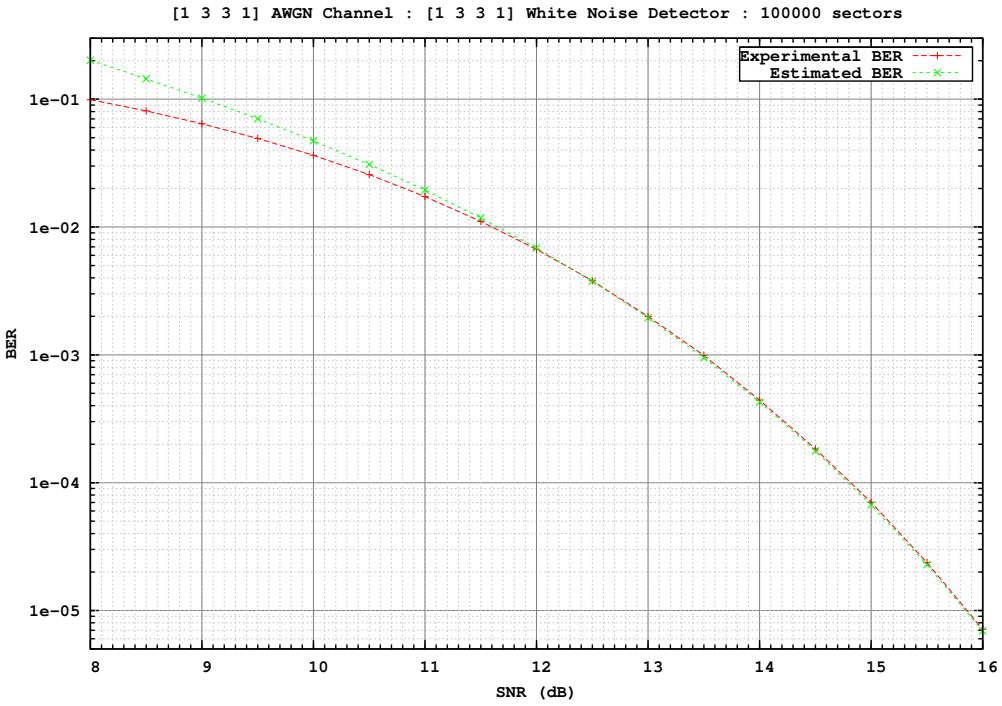


Figure 6.15: Estimated BER for white noise channel.

Next consider a realistic jitter dominated channel as described in (4.1.1) with 90% media noise, where the media noise consists of 90% position jitter and 10% phase jitter, and the error function describes the isolated transitions.

The plot shows the BER as predicted by the cost function against the actual

BER as measured for numerical simulation for white noise detectors with targets [1 3 3 1] and [2 6 3 -1].

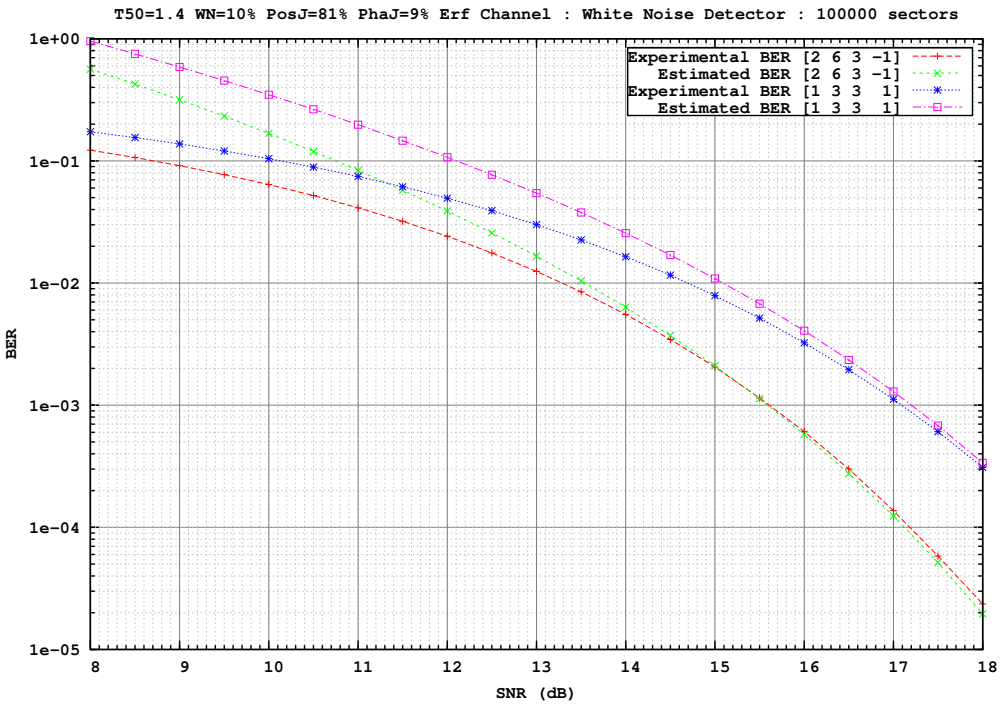


Figure 6.16: Estimated BER for jitter dominated channel with erf isolated transition.

In contract to the MMSE criteria, minimising BER does not whiten the noise. However observation of the spectrum reveals the noise is essentially white but with a low pass characteristic. Removing the high frequency noise seems intuitive since the most dangerous error events correspond to the rapidly alternating paths through the trellis.

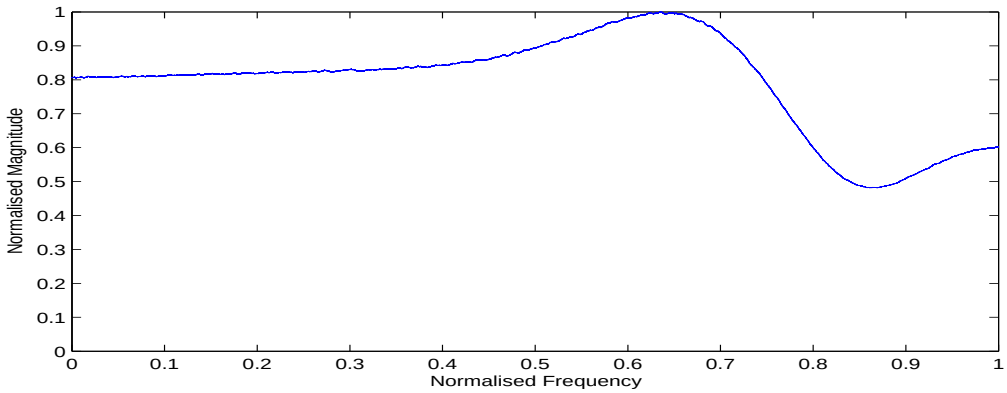


Figure 6.17: Spectrum of received signal equalised to target [1.00 2.06 1.13 -0.41] as chosen by BER criteria for 70% position jitter erf channel.

The following table shows that the target as selected by the BER criteria out performs the MMSE criteria.

Criteria	Target	Experimental SNR	BER
SNR	[1.00 2.21 2.21 1.00]	14.5 dB	2.23×10^{-2}
MMSE	[1.00 0.97 0.32 -0.04]	11.8 dB	6.98×10^{-3}
BER	[1.00 2.06 1.13 -0.41]	12.1 dB	6.29×10^{-3}

Table 6.4: Comparison of performance at 14.0 dB for 70% position jitter erf channel.

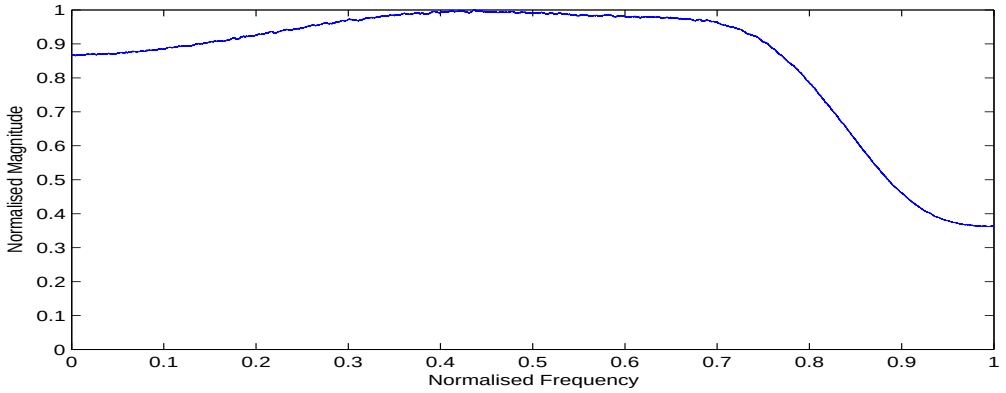


Figure 6.18: Spectrum of received signal equalised to target [1.00 1.11 0.12 -0.14] as chosen by BER criteria for 90% media noise 90% position jitter erf channel.

The following plot shows that the target as selected by the BER criteria out performs the MMSE criteria, and that the gap in dB is actually increasing as BER decreases.

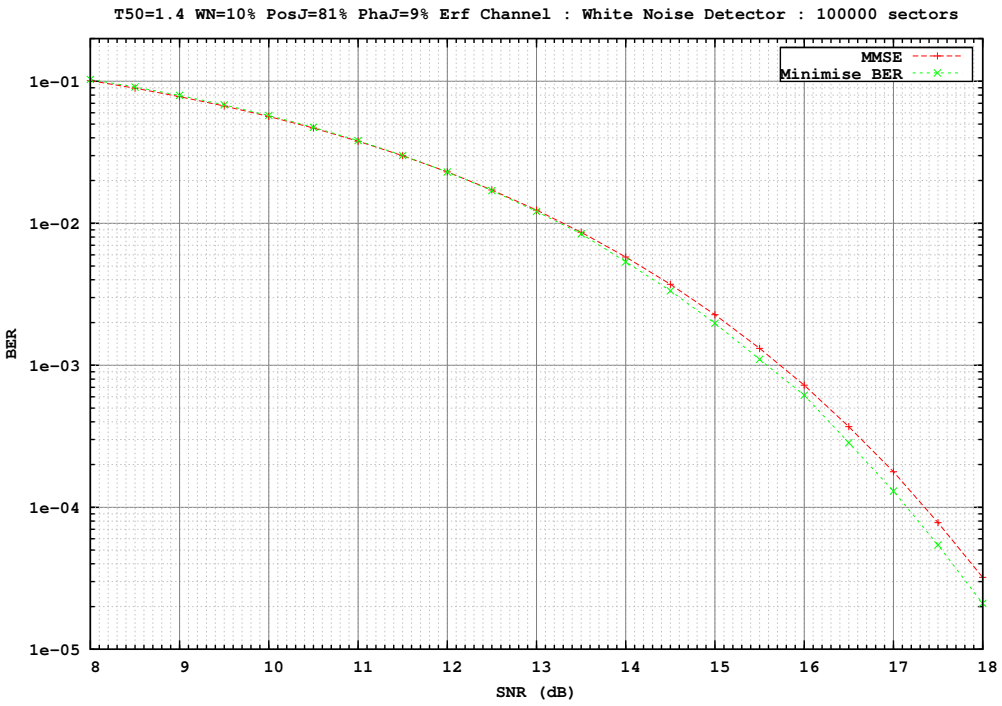


Figure 6.19: Comparison of performance for 90% media noise 90% position jitter erf channel.

The following plot shows the same trend again with a different isolated transition, showing that the BER criteria is resilient to different types of isolated transition.

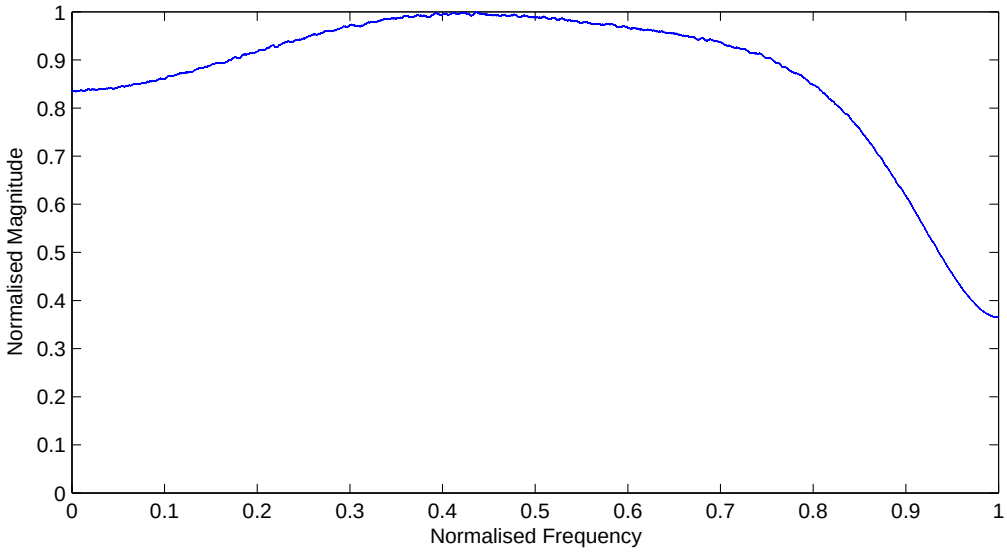


Figure 6.20: Spectrum of received signal equalised to target [1.00 2.97 1.13 -0.52] as chosen by BER criteria for 90% media noise 90% position jitter tanh channel.

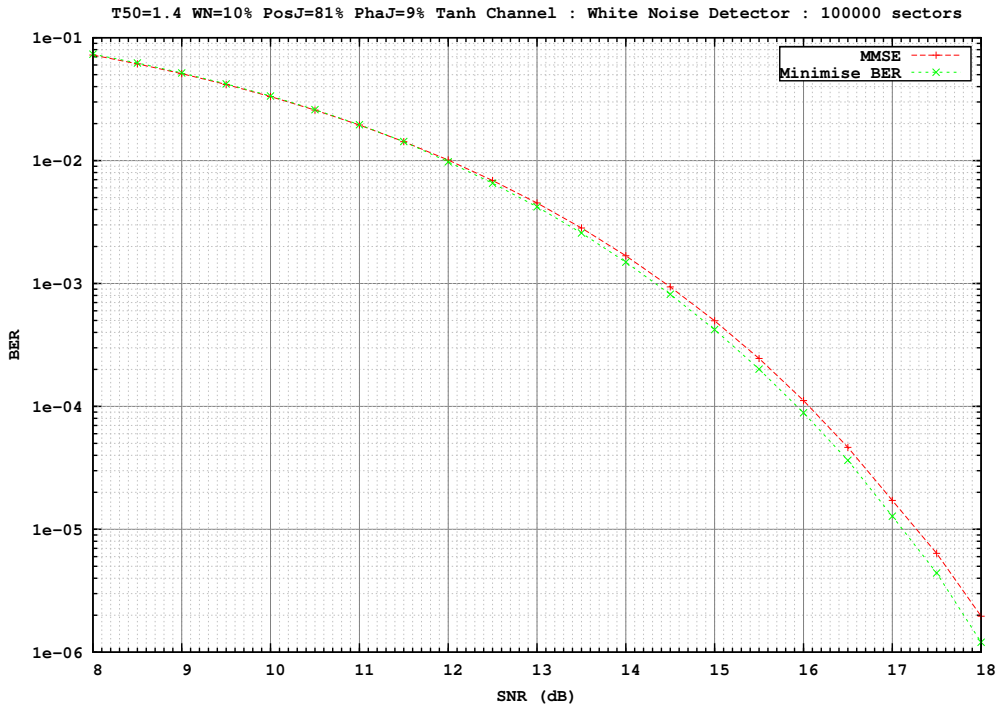


Figure 6.21: Comparison of performance for 90% media noise 90% position jitter tanh channel.

Chapter 7

Binary Addition

7.1 Introduction

When implementing a high performance Viterbi detector, we need a high speed implementation of a binary adder.

Binary addition is also a fundamental operation in computer arithmetic, and can be the limiting factor in determining the performance of processors, such as general purpose processors (CPU), 3D graphics processors (GPU), or digital signal processors (DSP).

Often designs can be pipelined by splitting the logic into multiple clock stages. This makes performance less of an issue, since additional pipeline stages can be added to meet performance requirements, as adding states reduces the logic per stage.

But when the result of the previous operation is required for the present

operation, this feedback loop prevents pipelining.

We see such a feedback loop in the ACS unit of Viterbi detectors, where path metrics must be added to the branch metrics, before a comparison selects the shortest path.

Another example is the address generation unit of a processor, where an offset determined by the current instruction is added to the address pointer which indexes the next instruction.

7.2 Background

An n-bit binary adder takes two n-bit binary numbers, a, b, and produces their $(n+1)$-bit binary sum, s.

For example, an 8-bit binary adder takes two 8-bit binary (0-255 decimal) numbers, say 175 & 178, and produces their 9-bit binary (0-511 decimal) sum, in this case 353.

$$\begin{array}{cccccccccc} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (178) \\ + & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & (175) \\ \hline 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & (353) \end{array}$$

We can represent this in general as follows:

$$\begin{array}{cccccccccc} & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ + & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ \hline s_8 & s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 \end{array}$$

We must find expressions for each $\{s_i\}$ in terms of the inputs $\{a_i\}$ and $\{b_i\}$.

s_0 can be determined by considering the following 4 cases

$$s_0 = \begin{cases} 0 & \text{if } a_0 = 0 \text{ and } b_0 = 0 \\ 1 & \text{if } a_0 = 0 \text{ and } b_0 = 1 \\ 1 & \text{if } a_0 = 1 \text{ and } b_0 = 0 \\ 0 & \text{if } a_0 = 1 \text{ and } b_0 = 1 \end{cases} \tag{7.2.1}$$

Therefore s_0 is 1 if exactly one of a_0 and b_0 is high (but not both). This is known in logic as an "exclusive-OR", and is represented as

$$s_0 = a_0 \oplus b_0 \tag{7.2.2}$$

In the case when both a_0 and b_0 are 1, a carry is generated in the second column. Therefore the carry, c_1, is given by the logical "AND" of a_0 and b_0, and is represented as

$$c_1 = a_0 b_0 \tag{7.2.3}$$

This reduces the problem to the following

$$\begin{array}{cccccccccc} & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & \\ + & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & \\ + & & & & & & & c_1 & \\ \hline s_8 & s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 \end{array}$$

We can now perform a similar consideration for the second column, and find

expressions for the sum and carry

$$\begin{aligned} s_1 &= a_1 \oplus b_1 \oplus c_1 \\ c_2 &= a_1 b_1 + c_1(a_1 + b_1) \end{aligned} \tag{7.2.4}$$

where + represents logical "OR".

This leads immediately to the generalisation, for $i \geq 0$

$$\begin{aligned} s_i &= a_i \oplus b_i \oplus c_i \\ c_{i+1} &= a_i b_i + c_i(a_i + b_i) \end{aligned} \tag{7.2.5}$$

7.3 Ripple Carry Adder

The circuit which implements the above method comprises n logic units connected in series, and is known as a ripple carry adder.

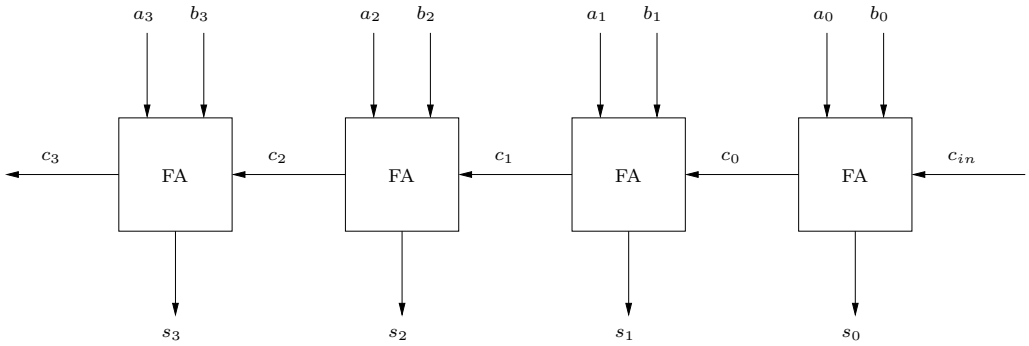


Figure 7.1: Ripple carry adder.

Ripple carry adders are ideal for small input sizes as the number of nodes in the prefix graph is minimised, which should result in a small silicon area.

Unfortunately the longest path through the a ripple carry adder circuit (known as the critical path), passes through n logic units, and therefore the delay through a ripple carry adder increases linearly with the input width. This makes them impractically slow for most applications, and other methods with lower delay complexity must be sought.

7.4 Carry Select Adder

Carry select adders [48] reduce the delay through the circuit, at the expense of increased silicon area, by using the divide and conquer technique.

The "divide" step of a carry select adder separates the most significant bits and the least significant bits into two separate groups.

$$\begin{array}{cccccc|cccc} & a_7 & a_6 & a_5 & a_4 & & a_3 & a_2 & a_1 & a_0 \\ + & b_7 & b_6 & b_5 & b_4 & & b_3 & b_2 & b_1 & b_0 \end{array}$$

The sum of the least significant bits is computed as before

$$\begin{array}{cccccc} & & a_3 & a_2 & a_1 & a_0 \\ & + & b_3 & b_2 & b_1 & b_0 \\ \hline & c_4 & s_3 & s_2 & s_1 & s_0 \end{array}$$

The sum of the most significant bits is computed in parallel with the LSB's, but in duplicate for the two possible cases where the carry out from the least significant bits, c_4, is 0 or 1.

$$\begin{array}{cccccc} & a_7 & a_6 & a_5 & a_4 \\ + & b_7 & b_6 & b_5 & b_4 \\ + & & & & 0 \\ \hline s_8^{(0)} & s_7^{(0)} & s_6^{(0)} & s_5^{(0)} & s_4^{(0)} \end{array} \qquad \begin{array}{cccccc} & a_7 & a_6 & a_5 & a_4 \\ + & b_7 & b_6 & b_5 & b_4 \\ + & & & & 1 \\ \hline s_8^{(1)} & s_7^{(1)} & s_6^{(1)} & s_5^{(1)} & s_4^{(1)} \end{array}$$

After all the sums have been computed, Either $s^{(0)}$ or $s^{(1)}$ is selected depending on the value of c_4.

The number of logic levels required is half that of a ripple carry adder to compute the separate sums, plus a single level to select the correct result.

The "conquer" step of a carry select adder now repeats the "divide" step for each of the smaller adders, further reducing the size of the adders by half, whilst only adding a single logic level to select the result.

This can be repeated recursively $\lceil \log_2 n \rceil$ times, at which point the adders will be trivial 1-bit adders, and there will be $\lceil \log_2 n \rceil$ logic levels of selections plus a single level for the 1-bit addition, giving a total of $\lceil \log_2 n \rceil + 1$ logic levels. For example a 64-bit carry select adder requires 7 logic levels, whilst a 64-bit ripple carry adders requires 64 logic levels.

7.5 Parallel Prefix Adder

Parallel prefix adders [49, 50] (or variations such as the Ling adder [51] described in section 7.7) are considered state of the art, as they have delay proportional to the logarithm of the input width, as per the carry select adder, but each individual logic level consists of simpler logic, thereby reduc-

ing overall delay, and they have a smaller silicon area.

Parallel prefix adders construct the carry from the inputs in a binary tree, whilst the ripple carry adder constructs the carry serially. Recall

$$\begin{aligned} s_i &= a_i \oplus b_i \oplus c_i \\ c_0 &= 0 \\ c_{i+1} &= a_i b_i + c_i(a_i + b_i) \qquad \text{for } i \geq 0 \end{aligned} \tag{7.5.1}$$

To simplify the equations, we introduce the generate and propagate functions

$$\begin{aligned} g_i &= a_i b_i \\ p_i &= a_i + b_i \end{aligned} \tag{7.5.2}$$

which can be interpreted as follows. A carry is definitely generated into the next column if both a_i AND b_i are 1. If a carry is received from the previous column, then that carry will propagate into the next column if either a_i OR b_i are 1. Therefore

$$c_{i+1} = g_i + p_i c_i \tag{7.5.3}$$

We can apply this formula recursively to find an expression for the carry into

column 4.

$$\begin{aligned} c_4 &= g_3 + p_3 c_3 \\ &= g_3 + p_3(g_2 + p_2 c_2) \\ &= g_3 + p_3 g_2 + p_3 p_2 c_2 \\ &= g_3 + p_3 g_2 + p_3 p_2 (g_1 + p_1 c_0) \\ &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 c_1 \\ &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 (g_0 + p_0 c_0) \\ &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \end{aligned} \tag{7.5.4}$$

which we will refer to as the group generate function from columns 3 to 0, and denote $G_{3:0} = c_4$. In general

$$G_{k-1:0} = c_k = g_{k-1} + p_{k-1} g_{k-2} + p_{k-1} p_{k-2} g_{k-3} + \ldots + p_{k-1} p_{k-2} p_{k-3} \ldots p_1 g_0 \tag{7.5.5}$$

Now let us now define the group propagate function $P_{k-1:0}$ to be

$$P_{k-1:0} = p_{k-1} p_{k-2} p_{k-3} \ldots p_0 \tag{7.5.6}$$

This give us a framework to construct carries in a systematic way. For example, the carry into column 4, $c_4 = G_{3:0}$

$$\begin{aligned} G_{3:0} &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \\ &= (g_3 + p_3 g_2) + (p_3 p_2)(g_1 + p_1 g_0) \\ &= G_{3:2} + P_{3:2} G_{1:0} \end{aligned} \tag{7.5.7}$$

where $G_{1:0}$, $G_{3:2}$ and $P_{3:2}$ are produced in parallel as

$$\begin{aligned} G_{1:0} &= g_1 + p_1 g_0 \\ G_{3:2} &= g_3 + p_3 g_2 \\ P_{3:2} &= p_3 p_2 \end{aligned} \tag{7.5.8}$$

This allows us to construct each carry in a binary tree. The first level of the tree computes the bit level generate and propagate functions directly from the inputs

$$\begin{aligned} g_i &= a_i b_i \\ p_i &= a_i + b_i \end{aligned} \tag{7.5.9}$$

The subsequent $\lceil \log_2 n \rceil$ levels of the tree produce increasingly larger group generate and group propagate functions by applying the relationship

$$\begin{aligned} G &= G_1 + P_1 G_0 \\ P &= P_1 P_0 \end{aligned} \tag{7.5.10}$$

To complete the adder, the sum is computed as $s_i = (a_i \oplus b_i) \oplus G_{i-1:0}$.

As a rough measure of delay complexity, we can count the number of logic gates on the critical path, and weight each gate with an approximate delay. We will count two input AND/OR logic gates as 2 unit delays, and XOR logic gates as 4 unit delays (since $a \oplus b = a\overline{b} + \overline{a}b$).

The circuit for the first level of the binary tree comprises either an AND for the generate or an OR for the propagate, both of which are 2 unit delays. For subsequent levels of the tree, the generate function requires an AND followed

by an OR, whilst the propagate function only requires an AND. Therefore the critical path is through the generate function and is 4 unit delays. The final XOR is 4 unit delays.

So the delay complexity for a parallel prefix adder is

$$PP(n) = 2 + 4\lceil \log_2 n \rceil + 4 = 4\lceil \log_2 n \rceil + 6 \tag{7.5.11}$$

For example

$$PP(64) = 30 \tag{7.5.12}$$

The measure of complexity is an over simplification for measuring the performance of electronic circuits, as it omits several important factors such as fanout, wire lengths and differences in capacitance between each type of logic gate.

For a more rigorous approach to measuring the complexity of adders see [52].

We have only discussed how to implement the carry logic for sizes which are powers of 2. Part of the art on constructing an efficient adder is deciding how best to implement the other carries, in terms of bit groupings and logic sharing.

For example Ladner-Fischer adders [50] construct lower order carries by padding the most significant bits with zeros. This results in the greatest logic sharing, and consequently the smallest implementation, but also results in the greatest fanout which slows down the adder.

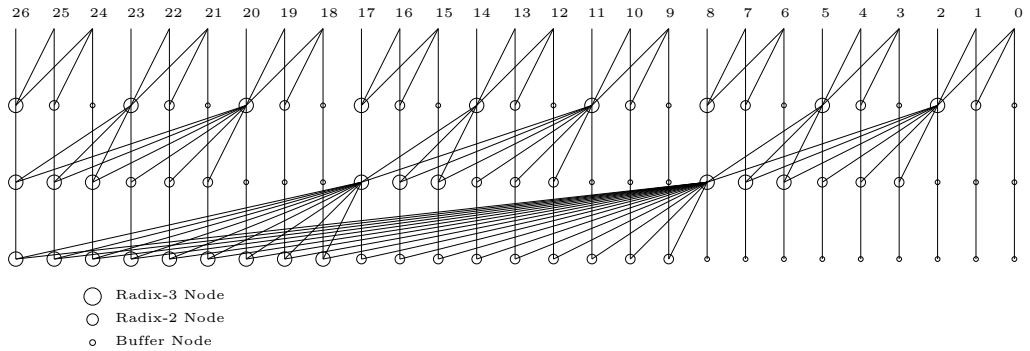


Figure 7.2: 27-bit Radix-3 Ladner-Fischer Adder.

By contrast, Kogge-Stone adders [49] construct lower order carries by padding the least significant bits with zeros, and re-indexing each of the inputs. This results in the least logic sharing and largest area implementation. It is however often the fastest implementation, as fanout is constant at each logic level.

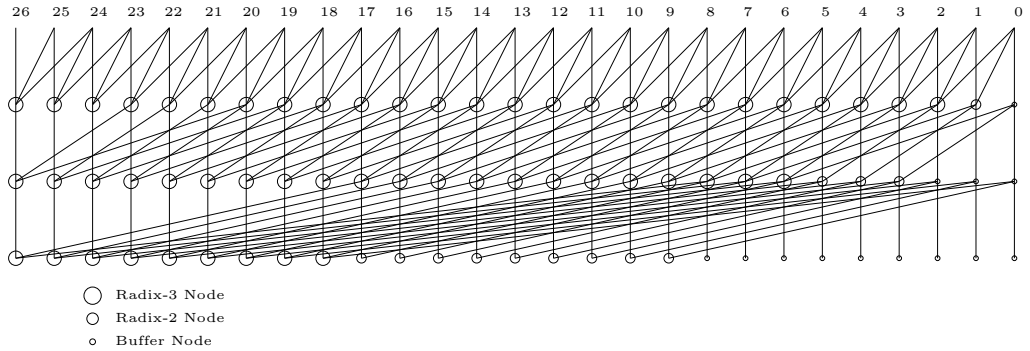


Figure 7.3: 27-bit Radix-3 Kogge-Stone Adder.

Knowles' family of adders [53] describe various trade-offs between Ladner-Fischer and Kogge-Stone adders.

Han-Carlson adders [54] and Brent-Kung adders [55] describe parallel prefix adders which are not of minimal depth, but offer good area trade-offs.

Other related architectures include carry-lookahead [56] and conditional sum [57].

However, none of the above describe fundamental differences to the underlying logic used to implement a single carry.

In this thesis, we shall restrict ourselves to considering only the underlying logic equations, and it will therefore suffice to use our simplified complexity model.

7.6 High Radix Parallel Prefix Adder

It is possible to build high radix trees, such as ternary trees, which combine 3 group generate functions per level. The larger group generate and propagate functions are formed as follows

$$\begin{aligned} G &= G_2 + P_2 G_1 + P_2 P_1 G_0 \\ P &= P_2 P_1 P_0 \end{aligned} \tag{7.6.1}$$

m-ary trees have the advantage of less logic levels ($\lceil \log_m n \rceil$), but each level is more complex.

$$\begin{aligned} G &= G_{m-1} + P_{m-1} G_{m-2} + \ldots + P_{m-1} \ldots P_1 G_0 \\ P &= P_{m-1} \ldots P_0 \end{aligned} \tag{7.6.2}$$

The critical path through the generate function is a m input AND gate, followed by an m input OR gate.

If we assume the delay through an m input AND/OR gate is m unit delays, then the delay complexity for a radix-m parallel prefix adder is

$$PP_m(n) = 2 + 2m\lceil \log_m n \rceil + 4 = 2m\lceil \log_m n \rceil + 6 \tag{7.6.3}$$

For example

$$PP_4(64) = 8 \log_4 64 + 6 = 30 \tag{7.6.4}$$

which is exactly the same as the standard radix-2 parallel prefix adder.

Therefore high radix parallel prefix adders are generally not used, as they offer no particular advantage of a radix-2 parallel prefix adder.

7.7 Ling Adder

Ling adders [51] do however offer a small advantage over parallel prefix adders.

Ling adders take advantage of the following identity

$$p_i g_i = (a_i + b_i)a_i b_i = a_i a_i b_i + b_i a_i b_i = a_i b_i + a_i b_i = a_i b_i = g_i \tag{7.7.1}$$

Therefore

$$\begin{aligned} G_{3:0} &= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \\ &= p_3 g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \\ &= p_3(g_3 + g_2 + p_2 g_1 + p_2 p_1 g_0) \\ &= p_3(g_3 + G_{2:0}) \end{aligned} \tag{7.7.2}$$

We define the pseudo generate function as

$$H_{k-1:0} = g_{k-1} + G_{k-2:0} \tag{7.7.3}$$

then

$$G_{k-1:0} = p_{k-1} H_{k-1:0} \tag{7.7.4}$$

Note that the pseudo generate function, H, can be formed in a binary tree similar to the standard generate function. The only difference is the indices on the propagate function are one less than for the parallel prefix recursion. For example the standard parallel prefix generate function is given by

$$\begin{aligned} G_{15:0} &= G_{15:8} + P_{15:8} G_{7:0} \\ G_{7:0} &= G_{7:4} + P_{7:4} G_{3:0} \\ G_{3:0} &= G_{3:2} + P_{3:2} G_{1:0} \\ G_{1:0} &= g_1 + p_1 g_0 \end{aligned} \tag{7.7.5}$$

whilst the pseudo generate function is given by

$$\begin{aligned} H_{15:0} &= H_{15:8} + P_{14:7} H_{7:0} \\ H_{7:0} &= H_{7:4} + P_{6:3} H_{3:0} \\ H_{3:0} &= H_{3:2} + P_{2:1} H_{1:0} \\ H_{1:0} &= g_1 + p_0 g_0 \end{aligned} \tag{7.7.6}$$

As with the parallel prefix adder, every level of the tree has 4 unit delays, as it has 2 two input gates on the critical path.

But one level of the tree can be simplified by applying the identity from (7.7.1).

$$H_{1:0} = g_1 + p_0 g_0 = g_1 + g_0 \tag{7.7.7}$$

Therefore one level of the binary tree has only 2 unit delays as it comprises a single two input gate on the critical path.

Since p_{i-1} is factored out of the pseudo carry, it must be recombined when computing the sum. However this can be written as follows, which removes this calculation from the critical path.

$$\begin{aligned} s_i &= a_i \oplus b_i \oplus G_{i-1:0} \\ s_i &= a_i \oplus b_i \oplus p_{i-1} H_{i-1:0} \\ &= \begin{cases} a_i \oplus b_i & \text{if } H_{i-1:0} = 0 \\ a_i \oplus b_i \oplus p_{i-1} & \text{if } H_{i-1:0} = 1 \end{cases} \\ &= \overline{H_{i-1:0}} x_i + H_{i-1:0} y_i \end{aligned} \tag{7.7.8}$$

where $x_i = a_i \oplus b_i$ and $y_i = a_i \oplus b_i \oplus p_{i-1}$.

Therefore the computation of the final sum can still be done in 4 unit delays despite having to recombine the p_{i-1} term with the pseudo carry.

Hence the delay complexity for a Ling adder is

$$L(n) = 2 + 2 + 4(\lceil \log_2 n \rceil - 1) + 4 = 4\lceil \log_2 n \rceil + 4 \tag{7.7.9}$$

For example

$$L(64) = 28 \tag{7.7.10}$$

As with parallel prefix, high radix Ling adders have no advantage over radix-2 Ling adders.

7.8 Generalisation of Ling Adder

It is possible to further generalise the Ling adder in order to reduce the complexity of high radix adder architectures [2, 5].

Recall the identity (7.7.1)

$$p_i g_i = (a_i + b_i)a_i b_i = a_i a_i b_i + b_i a_i b_i = a_i b_i + a_i b_i = a_i b_i = g_i \tag{7.8.1}$$

The obvious generalisation of this identity does not hold since, for example

$$P_{1:0} G_{1:0} \neq G_{1:0} \tag{7.8.2}$$

Therefore we aim to establish a new function which does satisfy an identity analogous to identity (7.7.1).

7.8.1 Generalised Ling Fundamentals

In this subsection, we introduce the generalised Ling pseudo generate and hyper propagate functions, then prove some identities relating to these func-

tions.

We start by defining a function D which satisfies an identity analogous to identity (7.7.1).

Definition 7.8.1. Define the following function

$$D_{k-1:0} = G_{k-1:0} + P_{k-1:0} \tag{7.8.3}$$

Lemma 7.8.2. The function $D_{k-1:0}$ satisfies the following identity

$$D_{k-1:0} = G_{k-1:1} + P_{k-1:0} \tag{7.8.4}$$

Proof.

$$\begin{aligned} D_{k-1:0} &= G_{k-1:1} + P_{k-1:1}g_0 + P_{k-1:0} \\ &= G_{k-1:1} + P_{k-1:1}g_0 + P_{k-1:1}p_0 \\ &= G_{k-1:1} + P_{k-1:1}(g_0 + p_0) \\ &= G_{k-1:1} + P_{k-1:1}p_0 \\ &= G_{k-1:1} + P_{k-1:0} \end{aligned} \tag{7.8.5}$$

□

Lemma 7.8.3. The following identity holds for the function D

$$D_{k-1:0}G_{k-1:0} = G_{k-1:0} \tag{7.8.6}$$

Proof.

$$\begin{aligned} D_{k-1:0}G_{k-1:0} &= (G_{k-1:1} + P_{k-1:0})G_{k-1:0} \\ &= (G_{k-1:1} + P_{k-1:1}p_0)(G_{k-1:1} + P_{k-1:1}g_0) \\ &= G_{k-1:1} + G_{k-1:1}P_{k-1:1}(p_0 + g_0) + P_{k-1:1}p_0g_0 \\ &= G_{k-1:1} + G_{k-1:1}P_{k-1:1}p_0 + P_{k-1:1}g_0 \\ &= G_{k-1:1} + G_{k-1:1}P_{k-1:0} + P_{k-1:1}g_0 \\ &= G_{k-1:1}(1 + P_{k-1:0}) + P_{k-1:1}g_0 \\ &= G_{k-1:1} + P_{k-1:1}g_0 \\ &= G_{k-1:0} \end{aligned} \tag{7.8.7}$$

□

We now have a function analogous to (7.7.1). We can now show that the function D can be used interchangeably with the propagate function P.

Lemma 7.8.4. *D satisfies the following identity*

$$G_1 + D_1G_0 = G_1 + P_1G_0 \tag{7.8.8}$$

Therefore D can be used as an alternative propagate function to P.

Proof.

$$\begin{aligned} G_1 + D_1G_0 &= G_1 + (G_1 + P_1)G_0 \\ &= G_1 + G_1G_0 + P_1G_0 \\ &= G_1(1 + G_0) + P_1G_0 \\ &= G_1 + P_1G_0 \end{aligned} \tag{7.8.9}$$

□

Note also that the function D has a decomposition which is analogous to the generate function G.

Lemma 7.8.5. Function D has the following decomposition

$$D = G_1 + P_1 D_0 \tag{7.8.10}$$

Proof.

$$\begin{aligned} D &= G + P \\ &= (G_1 + P_1 G_0) + (P_1 P_0) \\ &= G_1 + P_1(G_0 + P_0) \\ &= G_1 + P_1 D_0 \end{aligned} \tag{7.8.11}$$

□

Now we introduce another function B.

Definition 7.8.6. Define the following function

$$B_{k-1:0} = g_{k-1} + \ldots + g_0 \tag{7.8.12}$$

The function B satisfies the following identity.

Lemma 7.8.7. The generate function can be factorised as follows

$$D_{k-1:0} B_{k-1:0} = G_{k-1:0} \tag{7.8.13}$$

Proof. True for $k = 1$ since

$$D_{0:0}B_{0:0} = (G_{0:0} + P_{0:0})B_{0:0} = (g_0 + p_0)g_0 = p_0g_0 = g_0 = G_{0:0} \tag{7.8.14}$$

Suppose $D_{k-2:0}B_{k-2:0} = G_{k-2:0}$, then

$$\begin{aligned} D_{k-1:0}B_{k-1:0} &= (g_{k-1} + p_{k-1}D_{k-2:0})(g_{k-1} + B_{k-2:0}) \\ &= g_{k-1} + g_{k-1}B_{k-2:0} + g_{k-1}p_{k-1}D_{k-2:0} + p_{k-1}D_{k-2:0}B_{k-2:0} \\ &= g_{k-1}(1 + B_{k-2:0} + p_{k-1}D_{k-2:0}) + p_{k-1}D_{k-2:0}B_{k-2:0} \\ &= g_{k-1} + p_{k-1}D_{k-2:0}B_{k-2:0} \\ &= g_{k-1} + p_{k-1}G_{k-2:0} \qquad \text{(by assumption)} \\ &= G_{k-1:0} \end{aligned} \tag{7.8.15}$$

□

Finally we have an identity involving both D and B.

Lemma 7.8.8. The D function can be factorised as follows

$$D_{k-1:0} = D_{k-1:k-m}(B_{k-1:k-m} + D_{k-m-1:0}) \tag{7.8.16}$$

Proof. Follows directly from the identities we proved above.

$$\begin{aligned}
&D_{k-1:k-m}(B_{k-1:k-m} + D_{k-m-1:0}) \\
&\quad = D_{k-1:k-m}B_{k-1:k-m} + D_{k-1:k-m}D_{k-m-1:0} \\
&\quad = G_{k-1:k-m} + D_{k-1:k-m}D_{k-m-1:0} \\
&\quad = G_{k-1:k-m} + (G_{k-1:k-m} + P_{k-1:k-m})D_{k-m-1:0} \\
&\quad = G_{k-1:k-m}(1 + D_{k-m-1:0}) + P_{k-1:k-m}D_{k-m-1:0} \\
&\quad = G_{k-1:k-m} + P_{k-1:k-m}D_{k-m-1:0} \\
&\quad = D_{k-1:0}
\end{aligned} \tag{7.8.17}$$

□

We are now ready to introduce the generalisation of the Ling pseudo generate function (7.7.3).

Definition 7.8.9. Define the generalised pseudo generate function as

$$R^{(m)}_{k-1:0} = B_{k-1:k-m} + G_{k-m-1:0} \tag{7.8.18}$$

For example

$$R^{(2)}_{3:0} = B_{3:2} + G_{1:0} = g_3 + g_2 + g_1 + p_1 g_0 \tag{7.8.19}$$

Remark 7.8.10. Note that the Ling pseudo generate function (7.7.3) is a special case of the generalised pseudo generate function

$$R^{(1)}_{k-1:0} = H_{k-1:0} \tag{7.8.20}$$

Lemma 7.8.11. The generate function can be factorised in terms of the generalised pseudo generate function as follows

$$G_{k-1:0} = D_{k-1:k-m} R^{(m)}_{k-1:0} \tag{7.8.21}$$

Proof.

$$\begin{aligned} D_{k-1:k-m} R^{(m)}_{k-1:0} &= D_{k-1:k-m}(B_{k-1:k-m} + G_{k-m-1:0}) \\ &= D_{k-1:k-m} B_{k-1:k-m} + D_{k-1:k-m} G_{k-m-1:0} \\ &= G_{k-1:k-m} + D_{k-1:k-m} G_{k-m-1:0} \\ &= G_{k-1:k-m} + P_{k-1:k-m} G_{k-m-1:0} \\ &= G_{k-1:0} \end{aligned} \tag{7.8.22}$$

□

We also require a generalisation of the propagate function.

Definition 7.8.12. Define the generalised hyper propagate function as

$$Q^{(m)}_{k-1:0} = P_{k-1:k-m} D_{k-m-1:0} \tag{7.8.23}$$

For example

$$Q^{(2)}_{3:0} = P_{3:2} D_{1:0} = p_3 + p_2 + g_1 + p_1 p_0 \tag{7.8.24}$$

Remark 7.8.13. Note that the standard propagate function (7.5.6) is a special case of the generalised pseudo generate function

$$Q^{(0)}_{k-1:0} = P_{k-1:0} \tag{7.8.25}$$

Remark 7.8.14. Note also that the generalised pseudo generate function $Q^{(1)}$ is also equivalent to the standard propagate function (7.5.6). This explains why Ling adders use the regular propagate function.

$$Q^{(1)}_{k-1:0} = P_{k-1:1}D_{0:0} = P_{k-1:1}p_0 = P_{k-1:0} \tag{7.8.26}$$

Lemma 7.8.15. The D function can be factorisation in terms of the generalised pseudo generate function as follows

$$D_{k-1:0} = G_{k-1:k-m} + Q^{(m)}_{k-1:0} \tag{7.8.27}$$

Proof.

$$\begin{aligned} G_{k-1:k-m} + Q^{(m)}_{k-1:0} &= G_{k-1:k-m} + P_{k-1:k-m}D_{k-m-1:0} \\ &= G_{k-1:k-m} + P_{k-1:k-m}(G_{k-m-1:0} + P_{k-m-1:0}) \\ &= (G_{k-1:k-m} + P_{k-1:k-m}G_{k-m-1:0}) + (P_{k-1:k-m}P_{k-m-1:0}) \\ &= G_{k-1:0} + P_{k-1:0} \\ &= D_{k-1:0} \end{aligned} \tag{7.8.28}$$

□

7.8.2 Modified Pseudo Generate and Hyper Propagate Functions

The pseudo generate and hyper propagate functions introduced in the previous section, are the simplest functions which satisfy the desired criteria. However, it may be useful to introduce redundant terms into the pseudo generate and hyper propagate functions to facilitate a more regular adder structure.

Therefore we introduce a class of modified pseudo generate and modified hyper propagate functions. But first we define inclusion for binary functions.

Definition 7.8.16. A binary function Y is included in a binary function Z if $Y = 1 \Rightarrow Z = 1$, and we denote this $Y \subseteq Z$.

Definition 7.8.17. Define the modified pseudo generate function to be

$$\tilde{R}^{(m)}_{k-1:0} = X_{k-1:k-m} + G_{k-m-1:0} \tag{7.8.29}$$

where $X_{k-1:k-m}$ is any function such that

$$G_{k-1:k-m} \subseteq X_{k-1:k-m} \subseteq B_{k-1:k-m} \tag{7.8.30}$$

Definition 7.8.18. Define the modified hyper propagate function to be

$$\tilde{Q}^{(m)}_{k-1:0} = Y_{k-1:k-m} D_{k-m-1:0} \tag{7.8.31}$$

where $Y_{k-1:k-m}$ is any function such that

$$P_{k-1:k-m} \subseteq Y_{k-1:k-m} \subseteq D_{k-1:k-m} \tag{7.8.32}$$

We now show that the modified pseudo generate and hyper propagate functions can be used as a direct replacement for the regular pseudo generate and hyper propagate functions.

Lemma 7.8.19. The generate function can be factorised in terms of the modified pseudo generate function as follows

$$G_{k-1:0} = D_{k-1:k-m}\tilde{R}^{(m)}_{k-1:0} \tag{7.8.33}$$

Proof. Note that in the following proof, we use simplified notation for the function ranges

$$\begin{gathered} D_1(G_1 + G_0) \subseteq D_1\tilde{R} \subseteq D_1(B_1 + G_0) \\ D_1G_1 + D_1G_0 \subseteq D_1\tilde{R} \subseteq D_1B_1 + D_1G_0 \\ G_1 + D_1G_0 \subseteq D_1\tilde{R} \subseteq G_1 + D_1G_0 \\ G \subseteq D_1\tilde{R} \subseteq G \end{gathered} \tag{7.8.34}$$

□

Lemma 7.8.20. The D function can be factorised in terms of the modified pseudo generate function as follows

$$D_{k-1:0} = G_{k-1:k-m} + \tilde{Q}^{(m)}_{k-1:0} \tag{7.8.35}$$

Proof. Note that in the following proof, we use simplified notation for the function ranges

$$\begin{gathered}
G_1 + P_1 D_0 \subseteq G_1 + \tilde{Q} \subseteq G_1 + D_1 D_0 \\
G_1 + P_1 D_0 \subseteq G_1 + \tilde{Q} \subseteq G_1 + (G_1 + P_1) D_0 \\
G_1 + P_1 D_0 \subseteq G_1 + \tilde{Q} \subseteq G_1 (1 + D_0) + P_1 D_0 \\
G_1 + P_1 D_0 \subseteq G_1 + \tilde{Q} \subseteq G_1 + P_1 D_0 \\
G_1 + P_1 (G_0 + P_0) \subseteq G_1 + \tilde{Q} \subseteq G_1 + P_1 (G_0 + P_0) \\
(G_1 + P_1 G_0) + (P_1 P_0) \subseteq G_1 + \tilde{Q} \subseteq (G_1 + P_1 G_0) + (P_1 P_0) \\
G + P \subseteq G_1 + \tilde{Q} \subseteq G + P \\
D \subseteq G_1 + \tilde{Q} \subseteq D
\end{gathered} \tag{7.8.36}$$

□

7.8.3 Generalised Ling Radix-3 Recursion

In this subsection, we establish a radix-3 recursion for both the generalised Ling pseudo generate and hyper propagate functions.

Theorem 7.8.21. The pseudo generate function can be generated using the following recurrence relation.

$$R^{(\frac{3^{k+1}-1}{2})}_{3^{k+1}-1:0} = B_{3^{k+1}-1:2\cdot 3^k} + R^{(\frac{3^k-1}{2})}_{2\cdot 3^k-1:3^k} + Q^{(\frac{3^k+1}{2})}_{\frac{3^{k+1}-1}{2}:\frac{3^k+1}{2}} R^{(\frac{3^k-1}{2})}_{3^k-1:0} \tag{7.8.37}$$

Proof. First check for $k = 0$.

$$\begin{aligned} & B_{3^{k+1}-1:2\cdot 3^k} + R^{(\frac{3^k-1}{2})}_{2\cdot 3^k-1:3^k} + Q^{(\frac{3^k+1}{2})}_{\frac{3^{k+1}-1}{2}:\frac{3^k+1}{2}} R^{(\frac{3^k-1}{2})}_{3^k-1:0} \\ &= B_{2:2} + R^{(0)}_{1:1} + Q^{(1)}_{1:1} R^{(0)}_{0:0} \\ &= g_2 + g_1 + p_1 g_0 \\ &= R^{(1)}_{2:0} \\ &= R^{(\frac{3^{k+1}-1}{2})}_{3^{k+1}-1:0} \end{aligned} \tag{7.8.38}$$

Now prove the general case. We start by factorising the middle R term from definition (7.8.18) and the Q term from definition (7.8.23).

$$\begin{aligned} T &= B_{3^{k+1}-1:2\cdot 3^k} + R^{(\frac{3^k-1}{2})}_{2\cdot 3^k-1:3^k} + Q^{(\frac{3^k+1}{2})}_{\frac{3^{k+1}-1}{2}:\frac{3^k+1}{2}} R^{(\frac{3^k-1}{2})}_{3^k-1:0} \\ &= B_{3^{k+1}-1:2\cdot 3^k} + \left(B_{2\cdot 3^k-1:\frac{3^{k+1}+1}{2}} + G_{\frac{3^{k+1}-1}{2}:3^k} \right) \\ &\quad + \left(P_{\frac{3^{k+1}-1}{2}:3^k} D_{3^k-1:\frac{3^k+1}{2}} \right) R^{(\frac{3^k-1}{2})}_{3^k-1:0} \end{aligned} \tag{7.8.39}$$

We can then apply the identity from (7.8.21), followed by the regular parallel prefix recursion.

$$\begin{aligned} T = & \left(B_{3^{k+1}-1:2\cdot 3^k} + B_{2\cdot 3^k-1:\frac{3^{k+1}+1}{2}} \right) + G_{\frac{3^{k+1}-1}{2}:3^k} \\ & + P_{\frac{3^{k+1}-1}{2}:3^k} \left(D_{3^k-1:\frac{3^k+1}{2}} R^{(\frac{3^k-1}{2})}_{3^k-1:0} \right) \end{aligned} \tag{7.8.40}$$

followed by the regular parallel prefix recursion, which gives us the result as

required.

$$\begin{aligned} T &= B_{3^{k+1}-1:\frac{3^{k+1}+1}{2}} + G_{\frac{3^{k+1}-1}{2}:3^k} + P_{\frac{3^{k+1}-1}{2}:3^k} G_{3^k-1:0} \\ &= B_{3^{k+1}-1:\frac{3^{k+1}+1}{2}} + G_{\frac{3^{k+1}-1}{2}:0} \\ &= R^{(\frac{3^{k+1}-1}{2})}_{3^{k+1}-1:0} \end{aligned} \tag{7.8.41}$$

□

Note that the above recursion involves B terms, which are not formed by the general recursion. However, since $G \subseteq R \subseteq B$ we can replace the B terms with R terms to form a modified pseudo generate function, as this makes the recursion more regular.

Corollary 7.8.22. The modified pseudo generate function can be generated using the following recurrence relation.

$$\tilde{R}^{(\frac{3^{k+1}-1}{2})}_{3^{k+1}-1:0} = R^{(\frac{3^k-1}{2})}_{3^{k+1}-1:2\cdot 3^k} + R^{(\frac{3^k-1}{2})}_{2\cdot 3^k-1:3^k} + Q^{(\frac{3^k+1}{2})}_{\frac{3^{k+1}-1}{2}:\frac{3^k+1}{2}} R^{(\frac{3^k-1}{2})}_{3^k-1:0} \tag{7.8.42}$$

We must also determine a recurrence relation for the hyper propagate function.

Theorem 7.8.23. The hyper propagate function can be generated using the following recurrence relation.

$$Q^{(\frac{3^{k+1}+1}{2})}_{3^{k+1}-1:0} = P_{3^{k+1}-1:2\cdot 3^k} Q^{(\frac{3^k+1}{2})}_{2\cdot 3^k-1:3^k} \left(R^{(\frac{3^k-1}{2})}_{\frac{3^{k+1}-1}{2}-1:\frac{3^k-1}{2}} + Q^{(\frac{3^k+1}{2})}_{3^k-1:0} \right) \tag{7.8.43}$$

Proof. First check for $k = 0$.

$$\begin{aligned} & P_{3^{k+1}-1:2\cdot 3^k} Q^{(\frac{3^k+1}{2})}_{2\cdot 3^k-1:3^k} \left(R^{(\frac{3^k-1}{2})}_{\frac{3^{k+1}-1}{2}-1:\frac{3^k-1}{2}} + Q^{(\frac{3^k+1}{2})}_{3^k-1:0} \right) \\ &= P_{2:2} Q^{(1)}_{1:1} \left(R^{(0)}_{0:0} + Q^{(1)}_{0:0} \right) \\ &= p_2 p_1 (g_0 + p_0) \\ &= p_2 p_1 p_0 \\ &= Q^{(2)}_{2:0} \\ &= Q^{(\frac{3^{k+1}+1}{2})}_{3^{k+1}-1:0} \end{aligned} \tag{7.8.44}$$

Now prove the general case. We start by factorising terms from their definitions.

$$\begin{aligned} T &= P_{3^{k+1}-1:2\cdot 3^k} Q^{(\frac{3^k+1}{2})}_{2\cdot 3^k-1:3^k} \left(R^{(\frac{3^k-1}{2})}_{\frac{3^{k+1}-1}{2}-1:\frac{3^k-1}{2}} + Q^{(\frac{3^k+1}{2})}_{3^k-1:0} \right) \\ &= P_{3^{k+1}-1:2\cdot 3^k} \left(P_{2\cdot 3^k-1:\frac{3^{k+1}-1}{2}} D_{\frac{3^{k+1}-1}{2}-1:3^k} \right) \left(\left(B_{\frac{3^{k+1}-1}{2}-1:3^k} + G_{3^k-1:\frac{3^k-1}{2}} \right) \right. \\ &\quad \left. + \left(P_{3^k-1:\frac{3^k-1}{2}} D_{\frac{3^k-1}{2}-1:0} \right) \right) \end{aligned} \tag{7.8.45}$$

We can then simplify as follows

$$\begin{aligned} T &= \left(P_{3^{k+1}-1:2\cdot 3^k} P_{2\cdot 3^k-1:\frac{3^{k+1}-1}{2}} \right) D_{\frac{3^{k+1}-1}{2}-1:3^k} \left(B_{\frac{3^{k+1}-1}{2}-1:3^k} \right. \\ &\quad \left. + \left(G_{3^k-1:\frac{3^k-1}{2}} + P_{3^k-1:\frac{3^k-1}{2}} D_{\frac{3^k-1}{2}-1:0} \right) \right) \\ &= P_{3^{k+1}-1:\frac{3^{k+1}-1}{2}} D_{\frac{3^{k+1}-1}{2}-1:3^k} \left(B_{\frac{3^{k+1}-1}{2}-1:3^k} + D_{3^k-1:0} \right) \end{aligned} \tag{7.8.46}$$

Now we can apply the identity (7.8.16) to obtain

$$\begin{aligned} T &= P_{3^{k+1}-1:\frac{3^{k+1}-1}{2}} D_{\frac{3^{k+1}-1}{2}-1:0} \\ &= Q^{(\frac{3^{k+1}+1}{2})}_{3^{k+1}-1:0} \end{aligned} \tag{7.8.47}$$

as required. □

Again note that the above recursion involves P terms, which are not formed by the general recursion. However, since $P \subseteq Q \subseteq D$ we can replace the P terms with Q terms to form a modified hyper propagate function, as this makes the recursion more regular.

Corollary 7.8.24. The modified hyper propagate function can be generated using the following recurrence relation.

$$\tilde{Q}^{(\frac{3^{k+1}+1}{2})}_{3^{k+1}-1:0} = Q^{(\frac{3^k+1}{2})}_{3^{k+1}-1:2\cdot 3^k} Q^{(\frac{3^k+1}{2})}_{2\cdot 3^k-1:3^k} \left(R^{(\frac{3^k-1}{2})}_{\frac{3^{k+1}-1}{2}-1:\frac{3^k-1}{2}} + Q^{(\frac{3^k+1}{2})}_{3^k-1:0} \right) \tag{7.8.48}$$

7.8.4 Generalised Ling Radix-3 Example

Let us consider the example of the 27-bit carry $G_{26:0}$.

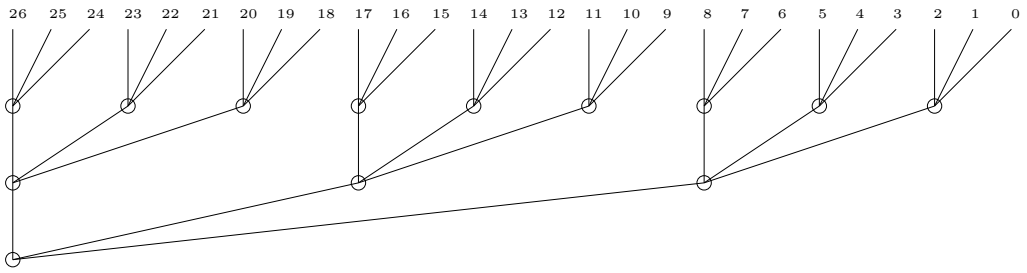


Figure 7.4: 27-bit Radix-3 Generalised Ling Carry.

At the first level of the adder we compute the bit level generate and propagate signals for $i = 0, \ldots 26$.

$$g_i = a_i b_i \quad p = a_i + b_i \tag{7.8.49}$$

At the second level of the adder, we compute the 3-bit group pseudo generate and hyper propagate functions

$$\begin{aligned}
R^{(1)}_{2:0} &= g_2 + g_1 + p_1 g_0 & Q^{(2)}_{4:2} &= p_4 p_3 p_2 \\
R^{(1)}_{5:3} &= g_5 + g_4 + p_4 g_3 & Q^{(2)}_{7:5} &= p_7 p_6 p_5 \\
R^{(1)}_{8:6} &= g_8 + g_7 + p_7 g_6 & Q^{(2)}_{10:8} &= p_{10} p_9 p_8 \\
R^{(1)}_{11:9} &= g_{11} + g_{10} + p_{10} g_9 & Q^{(2)}_{13:11} &= p_{13} p_{12} p_{11} \\
R^{(1)}_{14:12} &= g_{14} + g_{13} + p_{13} g_{12} & Q^{(2)}_{16:14} &= p_{16} p_{15} p_{14} \\
R^{(1)}_{17:15} &= g_{17} + g_{16} + p_{16} g_{15} & Q^{(2)}_{19:17} &= p_{19} p_{18} p_{17} \\
R^{(1)}_{20:18} &= g_{20} + g_{19} + p_{19} g_{18} & Q^{(2)}_{22:20} &= p_{22} p_{21} p_{20} \\
R^{(1)}_{23:21} &= g_{23} + g_{22} + p_{22} g_{21} & Q^{(2)}_{25:23} &= p_{25} p_{24} p_{23} \\
R^{(1)}_{26:24} &= g_{26} + g_{25} + p_{25} g_{24}
\end{aligned} \tag{7.8.50}$$

These are combined at the third level of the adder to produce 9-bit group

pseudo generate and hyper propagate functions

$$\begin{aligned}
R^{(4)}_{8:0} &= R^{(1)}_{8:6} + R^{(1)}_{5:3} + Q^{(2)}_{4:2}R^{(1)}_{2:0} \\
R^{(4)}_{17:9} &= R^{(1)}_{17:15} + R^{(1)}_{14:12} + Q^{(2)}_{13:11}R^{(1)}_{11:9} \\
R^{(4)}_{26:18} &= R^{(1)}_{26:24} + R^{(1)}_{23:21} + Q^{(2)}_{22:20}R^{(1)}_{20:18} \\
Q^{(5)}_{13:5} &= Q^{(2)}_{13:11}Q^{(2)}_{10:8}\left(R^{(1)}_{8:0} + Q^{(2)}_{7:5}\right) \\
Q^{(5)}_{22:14} &= Q^{(2)}_{22:20}Q^{(2)}_{19:17}\left(R^{(1)}_{17:9} + Q^{(2)}_{16:14}\right) \\
D_{26:23} &= p_{26}\left(R^{(1)}_{26:24} + Q^{(2)}_{25:23}\right)
\end{aligned} \tag{7.8.51}$$

Then the forth level produces the whole pseudo generate function

$$\begin{aligned}
R^{(13)}_{26:0} &= R^{(4)}_{26:18} + R^{(4)}_{17:9} + Q^{(5)}_{13:5}R^{(4)}_{8:0} \\
D_{26:14} &= D_{26:23}\left(R^{(4)}_{26:18} + Q^{(5)}_{22:14}\right)
\end{aligned} \tag{7.8.52}$$

The complete carry $G_{26:0}$ can be formed as

$$G_{26:0} = D_{26:14}R^{(13)}_{26:0} \tag{7.8.53}$$

Note that when constructing a sum bit of an adder, we can use the same technique as in (7.7.8) to move the above off the critical path.

7.8.5 Higher Radix Generalised Ling Recursions

As for the radix-3 case, we can construct recursions for other higher radix implementations as well.

For example, the radix-4 recursion has the form

$$\begin{aligned} R &= R_3 + R_2 + Q_2R_1 + Q_2Q_1R_0 \\ Q &= Q_3Q_2Q_1(R_1 + Q_0) \end{aligned} \tag{7.8.54}$$

It is also possible to use mixed radix implementation. For example, experiments have shown starting the first level with the standard radix-2 Ling implementation, then using higher radix generalised Ling for subsequent levels, yields an efficient implementation.

7.8.6 Generalised Ling Complexity

The critical path for each level of the radix-3 generalised Ling tree has the form

$$\begin{aligned} R &= R_2 + R_1 + Q_1R_0 \\ Q &= Q_2Q_1(R_1 + Q_0) \end{aligned} \tag{7.8.55}$$

Each of which can be implemented with a critical path comprising 2 two input gates in series, which means each level has 4 unit delays.

Therefore the delay complexity for a radix-3 generalised Ling adder is

$$GL_3(n) = 2 + 4\lceil \log_3 n \rceil + 4 = 4\lceil \log_3 n \rceil + 6 \tag{7.8.56}$$

Similarly, the critical path for each level of the radix-4 generalised Ling tree has the form

$$\begin{aligned} R &= R_3 + R_2 + Q_2R_1 + Q_2Q_1R_0 \\ Q &= Q_3Q_2Q_1(R_1 + Q_0) \end{aligned} \tag{7.8.57}$$

Each of which can be implemented with a critical path comprising 2 three input gates in series, which means each level has 6 unit delays.

Therefore the delay complexity for a radix-4 generalised Ling adder is

$$GL_4(n) = 2 + 6\lceil \log_4 n \rceil + 4 = 4\lceil \log_4 n \rceil + 6 \tag{7.8.58}$$

For example, let us consider a 64-bit radix-4 generalised Ling carry.

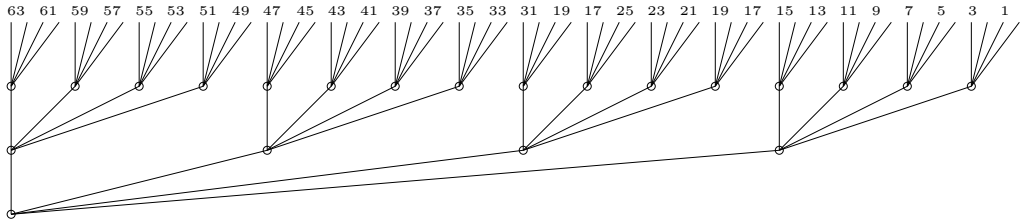


Figure 7.5: 64-bit Radix-4 Generalised Ling Carry.

By applying the above complexity formula we see that

$$GL_4(64) = 24 \tag{7.8.59}$$

7.9 Summary

Below is a comparison of the complexity of generalised Ling adders against other state of the art adder implementations.

Radix	Parallel Prefix	Complexity	64-bit Example
2	$G_1 + P_1G_0$	$4\lceil \log_2 n \rceil + 6$	30
3	$G_2 + P_2G_1 + P_2P_1G_0$	$6\lceil \log_3 n \rceil + 6$	30
4	$G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$	$8\lceil \log_4 n \rceil + 6$	30

Table 7.1: Complexity of parallel prefix adders.

Radix	Ling (first level only)	Complexity	64-bit Example
2	$H_1 + H_0$	$4\lceil \log_2 n \rceil + 4$	28
3	$H_2 + H_1 + P_1 H_0$	$6\lceil \log_3 n \rceil + 4$	28
4	$H_3 + H_2 + P_2 H_1 + P_2 P_1 H_0$	$8\lceil \log_4 n \rceil + 4$	28

Table 7.2: Complexity of Ling adders. Logic simplification of the first level is shown. Subsequent levels have the same structure as parallel prefix.

Radix	Generalised Ling	Complexity	64-bit Example
3	$R_2 + R_1 + Q_1 R_0$	$4\lceil \log_3 n \rceil + 6$	22
4	$R_3 + R_2 + Q_2 R_1 + Q_2 Q_1 R_0$	$6\lceil \log_4 n \rceil + 6$	24

Table 7.3: Complexity of generalised Ling adders.

We see that generalised Ling implementations compare favourably against both parallel prefix and Ling adders.

Chapter 8

Conclusions

In this chapter we summarise the main results from the thesis.

8.1 Loop Elimination

Loop elimination provides a novel approach to increasing the throughput of Viterbi detectors.

Figures 2.15 and 2.16 show how loop elimination can reduce the delay complexity of a 3T implementation to that of a 2T implementation, and how loop elimination can be recursively applied to a trellis, which allows the trellis to be decoded in a logarithmic number of parallel steps.

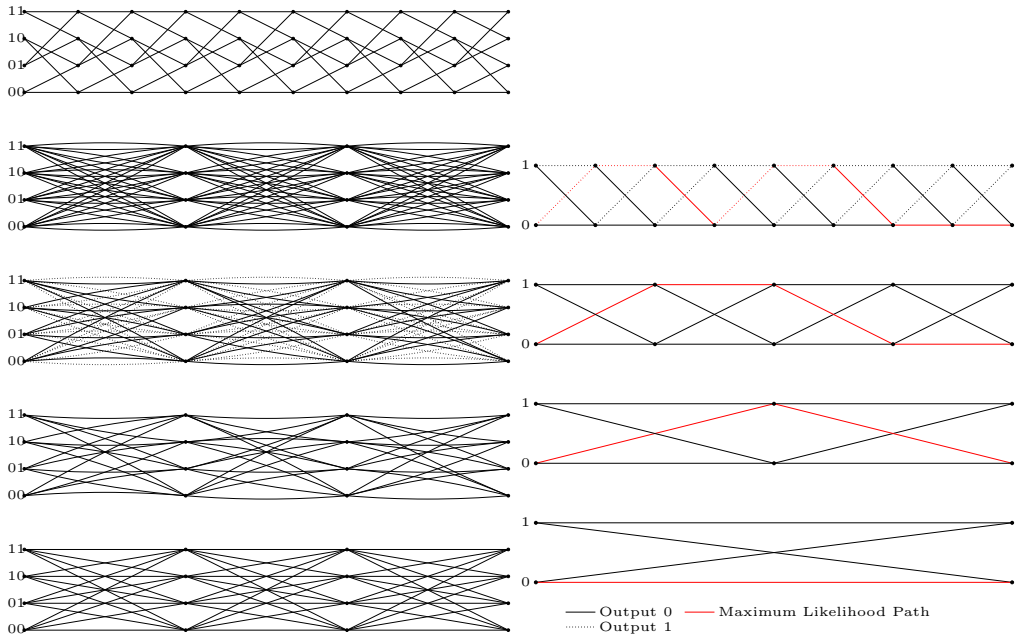


Figure 8.1: Loop elimination.

In section 2.4.1 we show that with loop elimination, the delay through a Viterbi detector remains constant for detectors whose radix, T, exceeds the constraint length, K.

This facilitates the implementation of detectors with high throughput, as the delay per output decreases inverse proportionally with the radix, whilst the standard implementation has an asymptotically constant delay per output.

Table 2.4 summarises the complexity of loop elimination.

Method	Delay	Delay/output	Total area
Trellis unrolling	$T+1$	$(T+1)/T$	$2^{T+1}-1$
Loop elimination	K	K/T	$2^{T}+2^{K-1}-1$

Table 8.1: Complexity comparison of trellis unrolling against loop elimination.

8.2 Invariants

Invariants provide a novel approach to reducing the complexity of Viterbi detectors.

In particular, we show that the path metric difference between the two sides of a loop is invariant in (2.6.6)

$$a_i - b_i = g_i \tag{8.2.1}$$

We also show in (2.6.12) that the four sides of two loops originating from the same initial state, have the following invariance

$$a_i + b_i - c_i - d_i = \begin{cases} \pm 2g_0 & \text{if } i = k-1 \\ 0 & \text{if } i = 0, \ldots, k-2 \end{cases} \tag{8.2.2}$$

and similarly in (2.6.15) we show that four sides of two loops originating from the same final state, have the following invariance

$$a_i + b_i - c_i - d_i = \begin{cases} \pm 2g_{i-1} & \text{if } i = 0, \ldots, k-2 \\ 0 & \text{if } i = k-1 \end{cases} \tag{8.2.3}$$

In section 2.7.3, we consider how these invariants can be applied to a 4-state 3T Viterbi detector implementation. Table 2.16 summarises the reduction in complexity when compared with a standard 2T implementation and a 3T implementation that uses loop elimination alone.

BMU Implementation	Adders	Multipliers	Outputs/Cycle
2T	35	6	2
3T + Loops	88	9	3
3T + Loops + Invariants	47	9	3

Table 8.2: Complexity of various 4-state implementations.

8.3 Data Dependent Double Detectors

Double detectors are an innovative approach to data dependent Viterbi detectors, whose performance closely matches the performance of auto-regressive and data dependent noise predictive detectors, but without the practical implementation drawbacks of high complexity or low throughput.

Figure 5.9 illustrates how the double detector replaces the noise predictive loop found in DDNP detectors, with survivor information from a pre-detector, thus avoiding the long feedback loop which results in low throughput.

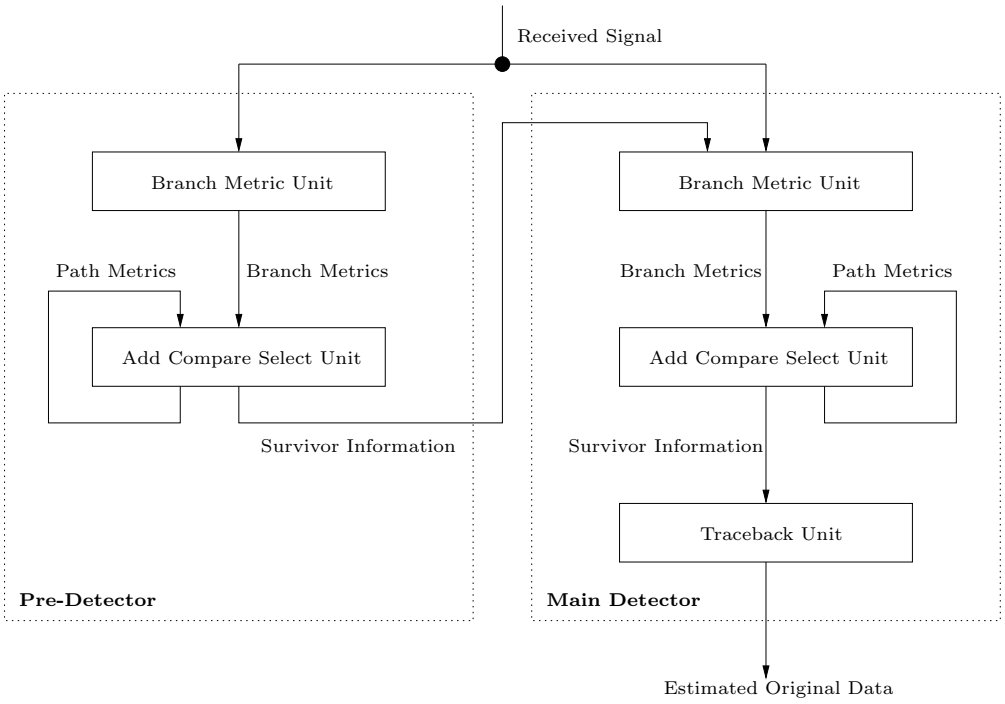


Figure 8.2: Double detector replaces noise predictive loop with survivor information from pre-detector.

Figure 5.10 shows simulation results comparing the performance of the double detector against auto-regressive and data dependent noise predictive detectors.

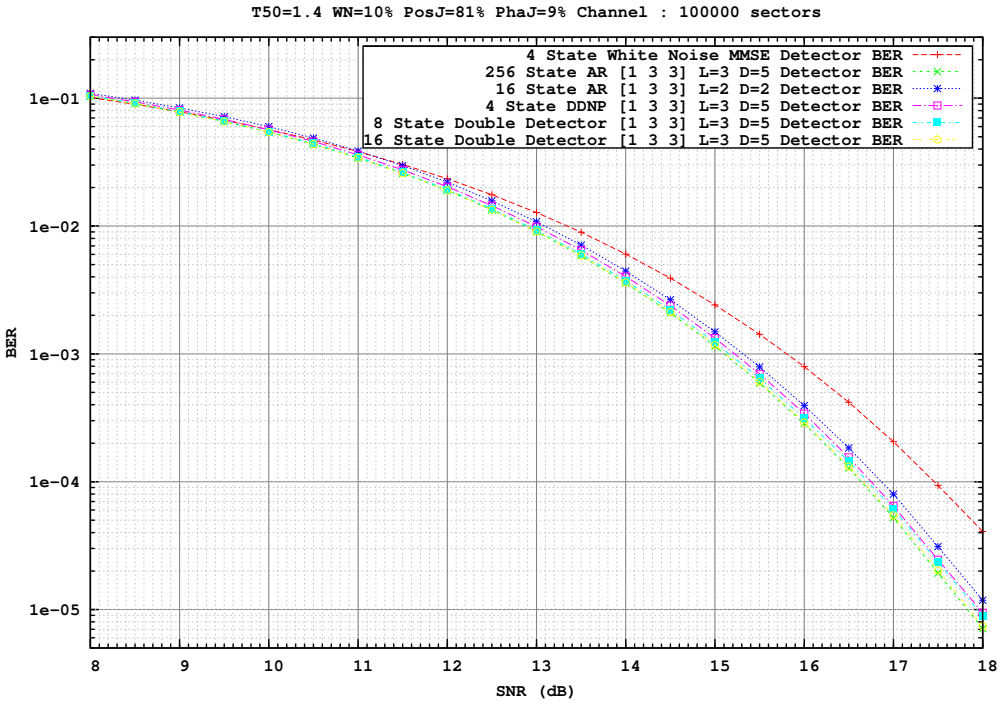


Figure 8.3: Comparison of white noise MMSE detector, data dependent autoregressive detector, data dependent noise predictive detector and double detector.

8.4 Cost Function

The cost function introduces a novel approach to analytically determine the bit error rate of a Viterbi detector from only the statistics of the channel.

In particular, in equation (6.1.36) we show that the bit error rate can be well approximated by the function

$$\sum_{\substack{\left\{P\in\mathcal{P}|P^{(j)}=x_j \text{ for } j<t-L \text{ and } j>t\right\} \\ P\neq\underline{x}}} W(P)\mathbb{P}(PM(P)\leq PM(\underline{x})\mid \underline{x},\underline{r})$$

where $W(P) = \left|\left\{j \mid P^{(j)} \neq x_j\right\}\right|$ is the weight of the path, and from (6.1.47) that

$$\mathbb{P}(PM_{error} < PM_{actual} \mid \underline{x}, \underline{r}) \quad = \frac{1}{2}\operatorname{erfc}\left(\frac{\mathbb{E}\left[PM_{error} - PM_{actual}\right]}{\sqrt{\operatorname{Var}\left[PM_{error} - PM_{actual}\right]}}\right)$$

Then in (6.1.63) and (6.1.64) we show that

$$\mathbb{E}\left[PM_{error} - PM_{actual}\right] \qquad \operatorname{Var}\left[PM_{error} - PM_{actual}\right]$$

can be expressed in terms of the following statistics of the channel

$$\mathbb{E}\left[r_{t-i}\right] \qquad \mathbb{E}\left[x_{t-i}r_{t-j}\right] \qquad \mathbb{E}\left[r_{t-i}r_{t-j}\right]$$

Figure 6.15 shows that at high SNR, the cost function accurately estimates bit error rate

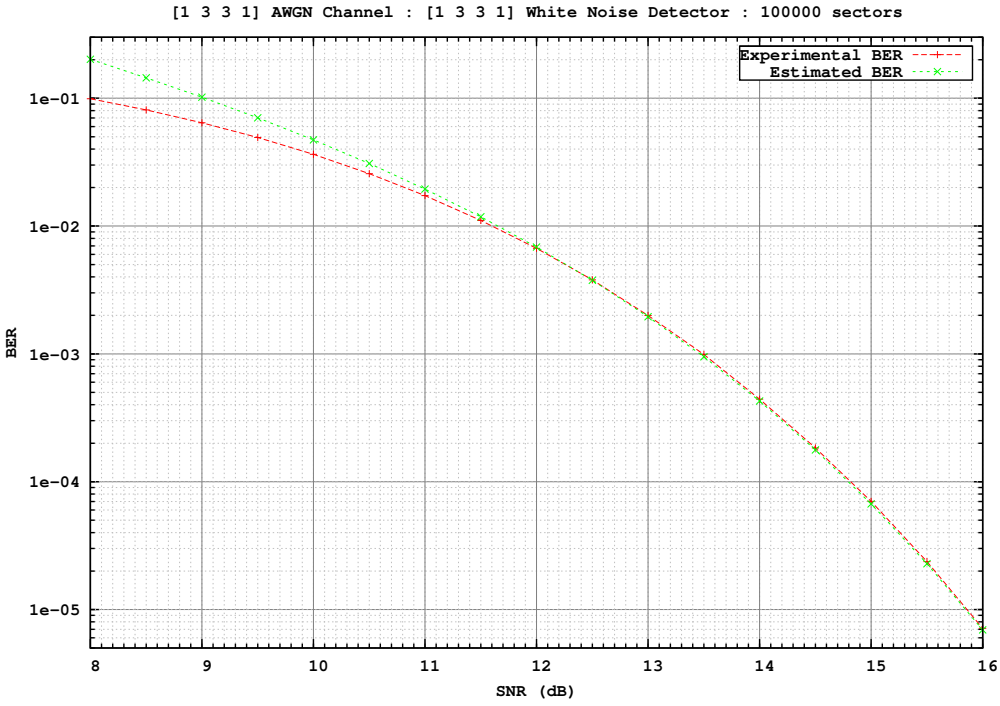


Figure 8.4: Estimated BER for white noise channel.

Finally we apply the cost function to the problem of determining ISI target and equaliser coefficients that minimise bit error rate, and show that the cost function out performs the MMSE criteria in figure 6.19.

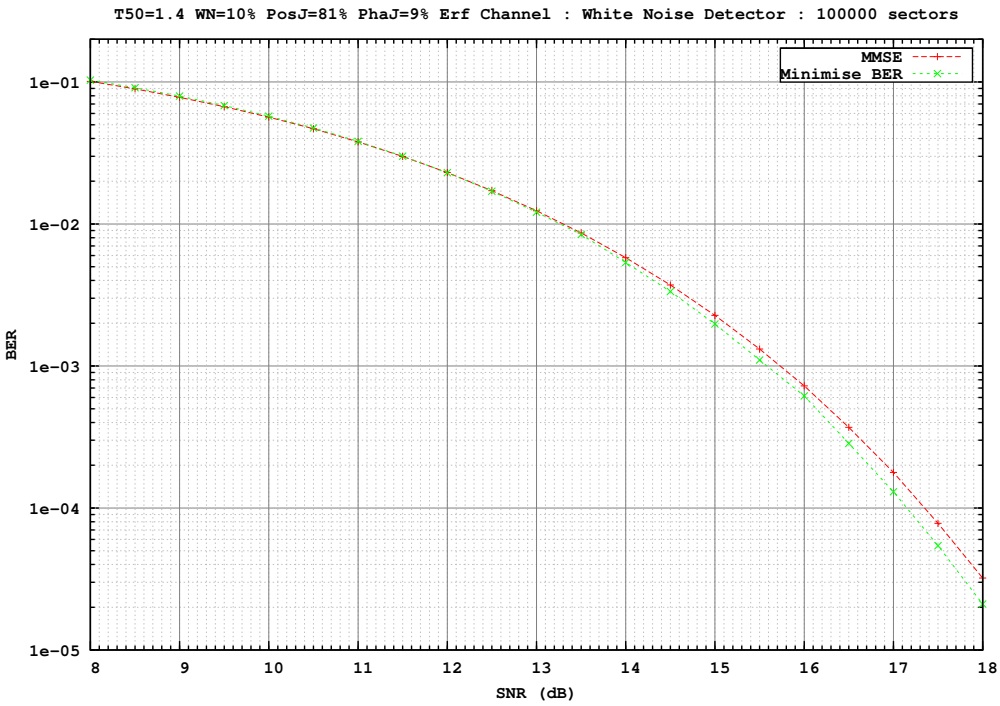


Figure 8.5: Comparison of performance for 90% media noise 90% position jitter erf channel.

8.5 Binary Addition

We introduce a novel approach to binary addition based on a generalisation of the Ling prefix adder.

In (7.8.55) and (7.8.57) we show that the recursion for radix-3 and radix-4 implementations has the following form

$$R = R_2 + R_1 + Q_1 R_0 \qquad Q = Q_2 Q_1 (R_1 + Q_0)$$

$$R = R_3 + R_2 + Q_2 R_1 + Q_2 Q_1 R_0 \quad Q = Q_3 Q_2 Q_1 (R_1 + Q_0)$$

and that the corresponding delay complexities (7.8.56) and (7.8.58)

$$GL_3(n) = 4\lceil \log_3 n \rceil + 6$$

$$GL_4(n) = 4\lceil \log_4 n \rceil + 6$$

compare favourably with the delay complexity of the standard Ling adder (7.7.9).

$$L(n) = 4\lceil \log_2 n \rceil + 4$$

Related References

[1] S. Gratrix, R.C. Jackson, T. Parnell, and O. Zaboronski. Viterbi detector for non-Markov recording channels. *IEEE Trans. Mag.*, 44(1):198–206, 2008.

[2] R.C. Jackson and S. Talwar. High speed binary addition. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference*, volume 2, pages 1350–1353. IEEE, November 2004.

[3] N. Atkinson, R.C. Jackson, O. Zaboronski, T. Drane, and A. Vityaev. Method and apparatus for an effective path metric computation in maximal likelihood decoding. *U.S. Patent Application*, app. 10/867,179, pub. US 2005/0094748 A1, June 2004.

[4] S. Gratrix, R.C. Jackson, and O. Zaboronski. Likelihood detector apparatus and method. *U.S. Patent Application*, app. 11/766,540, pub. US 2008/0002791 A1, June 2007.

[5] R.C. Jackson and S. Talwar. Logic circuit and method for carry and sum generation and method of designing such a logic circuit. *U.S. Patent*, app. 10/714,408, pub. US 2004/0153490 A1, iss. 7,260,595, November 2003.

[6] S. Gratrix, R.C. Jackson, and O. Zaboronski. Likelihood detector for data-dependent correlated noise. *U.S. Patent Application*, USP92288, 2007.

Bibliography

[7] R.C. Jackson, D. Rumynin, and O. Zaboronski. An approach to RAID-6 based on cyclic groups of a prime order. Submitted to Applicable Algebra in Engineering, Communication and Computing, [arXiv:cs/0611109v1].

[8] A.D. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13:260–269, 1967.

[9] G.D. Forney. The Viterbi algorithm. *IEEE Trans. Inform. Theory*, 61:268–278, 1973.

[10] H. Kobayashi and D.T. Tang. Application of partial-response channel coding to magnetic recording systems. *Bell J. Res. Develop.*, July 1970.

[11] A. Kavcic and A. Patapoutian. A signal-dependent auto-regressive channel model. *IEEE Trans. Mag.*, 35(5):2316–2318, September 1999.

[12] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions*, 53:370–418, 1763.

[13] S. Sridharan and L.R. Carley. A 110 MHz 350 mW 0.6 μm cmos 16-state generalized-target Viterbi detector for disk drive read channels. *IEEE Journal of Solid-State Circuits*, 35(3):362–370, March 2000.

[14] P. Black and T. Meng. A 140 Mb/s 32-state radix-4 Viterbi decoder. *IEEE Journal of Solid-State Circuits*, 27(12):1877–1885, December 1992.

[15] A.K. Yeung and J.M. Rabaey. A 210 Mb/s radix-4 bit-level pipelined Viterbi decoder. In *IEEE International Solid-State Circuits Conference, ISSCC'95, Digest of Technical Papers*, pages 88–89, San Francisco, CA, USA, February 1995.

[16] G. Fettweis, R. Karabed, P.H. Siegel, and H.K. Thapar. Reduced-complexity Viterbi detector architectures for partial response signaling. In *Proceedings IEEE Global Telecommunications Conference*, pages 559–563, Singapore, November 1995.

[17] E. Yeo, S. Augsburger, W.R. Davis, and B. Nikolic. A 500 Mb/s soft-output Viterbi decoder. *IEEE Journal of Solid-State Circuits*, 38(7):1234–1241, July 2003.

[18] A. Avizienis. Signed-digit representation for fast parallel arithmetic. *IRE Transactions on Computers*, EC-10:389–400, September 1961.

[19] G. Fettweis and H. Meyer. Parallel Viterbi algorithm: breaking the acs-bottleneck. *IEEE Transactions on Communications*, 37(8):785–790, August 1989.

[20] G. Fettweis and H. Meyer. High-rate Viterbi processor: a systolic array solution. *IEEE Journal on Selected Areas in Communications*, 8(8):1520–1534, October 1990.

[21] G. Fettweis and H. Meyer. High-speed parallel Viterbi decoding algorithm and vlsi architecture. *IEEE Communications Magazine*, 29(8):46–55, May 1991.

[22] IEEE Computer Society. *IEEE standard for information technology, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE, June 2007.

[23] IEEE Computer Society. *IEEE standard for local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems*. IEEE, October 2004.

[24] J. Moon. Performance comparison of detection methods in magnetic recording. *IEEE Trans. Mag.*, 26(6):3155–3172, 1990.

[25] J. Moon. Discrete-time modeling of transition-noise-dominated channels and study of detection performance. *IEEE Trans. Mag.*, 27(6):4573–4578, November 1991.

[26] N.R. Belk, P.K. George, and G.S. Mowry. Noise in high performance thin-film longitudinal magnetic recording media. *IEEE Trans. Mag.*, Mag-21(5):1350–1355, 1985.

[27] Z. Zhang, T.M. Duman, and E.M. Kurtas. Information rates of binary-input intersymbol interference channels with signal-dependent media noise. *IEEE Trans. Mag.*, 39(1):599–607, January 2003.

[28] J. Moon and L.R. Carley. Detection performance in the presence of transition noise. *IEEE Trans. Mag.*, 26(5):2172–2174, September 1990.

[29] J. Moon, L.R. Carley, and R.R. Katti. Density dependence of noise in thin metallic longitudinal media. *J. Appl. Phys.*, 63:3254, 1988.

[30] T.C. Arnoldussen and H.C. Tong. Zigzag transition profiles, noise and correlation statistics in highly oriented longitudinal film media. *IEEE Trans. Mag.*, 22:889, 1986.

[31] W. Ryan. Optimal code rates for concatenated codes on a PR4-equalized magnetic recording channel. In *CISS*, 1999.

[32] J. Moon. SNR definition for magnetic recording channels with transition noise. *IEEE Trans. Mag.*, 36:3881, 2001.

[33] H.N. Bertram. *Theory of magnetic recording*, chapter 8. Cambridge University Press, 1994.

[34] T.C. Arnoldussen and L.L. Nunnelley. *Noise in digital magnetic recording.* World Scientific, 1992.

[35] M.F. Erden, I. Ozgunes, E.M. Kurtas, and W. Eppler. General transform filters in perpendicular recording architectures. *IEEE Trans. Mag.*, 38(5):2334–2336, September 2002.

[36] B. Valcu, T. Roscamp, and H.N. Bertram. Pulse shape, resolution, and signal-to-noise ratio in perpendicular recording. *IEEE Trans. Mag.*, 38:288, 2002.

[37] G.D. Forney. Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. *IEEE Trans. Inform. Theory*, IT-18(3):363–378, 1972.

[38] P.R. Chevillat, E. Eleftheriou, and D. Maiwald. Noise-predictive partial-response equalizers and applications. In *Proceedings of ICC*, pages 942–947. IEEE, 1992.

[39] J.D. Coker, E. Eleftheriou, R.L. Galbraith, and W. Hirt. Noise-predictive maximum likelihood (NPML) detection. *IEEE Trans. Mag.*, 34(1):110–117, 1998.

[40] J. Caroselli and J.K. Wolf. Applications of a new simulation model for media noise limited magnetic recording channels. *IEEE Trans. Mag.*, 32(5):3917–3919, September 1996.

[41] H.N. Bertram. *Theory of magnetic recording.* Cambridge University Press, 1994.

[42] T.R. Oenning and J. Moon. Modelling the Lorentzian magnetic recording channel with transition noise. *IEEE Trans. Mag.*, 37(1):583–591, January 2001.

[43] J. Caroselli, S.A. Alteker, P. McEwen, and J.K. Wolf. Improved detection for magnetic recording systems with media noise. *IEEE Trans. Mag.*, 33(5):2779–2781, 1997.

[44] A. Kavcic and J.M. Moura. The Viterbi algorithm and Markov noise memory. *IEEE Trans. Inform. Theory*, 46(1):291–301, 2000.

[45] J. Moon and J. Park. Pattern-dependent noise prediction in signal-dependent noise. *IEEE J. Select. Areas on Commn.*, 19(4):730–743, 2001.

[46] D.J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*, chapter 2, page 34. Cambridge University Press, 2003.

[47] J. Moon and W. Zeng. Equalization for maximum likelihood detectors. *IEEE Trans. Mag.*, 31(2):1083–1088, 1995.

[48] O.J. Bedrij. Carry select adder. *IRE Trans.*, EC-11:340–346, June 1962.

[49] P.M. Kogge and H.S. Stone. A parallel algorithm for efficient solution of a general class of recurrence equations. *IEEE Trans. Computers*, C-22(8):786–793, August 1973.

[50] R.E. Ladner and M.J. Fischer. Parallel prefix computation. *Journal of ACM*, 27(4):831–838, October 1980.

[51] H. Ling. High speed binary adder. *IBM Journal of Research and Development*, 25(3):156–166, 1981.

[52] N. Burgess. New models of prefix adder topologies. *Journal of VLSI Signal Processing Systems*, 40(1):125–141, May 2005.

[53] S. Knowles. A family of adders. In *Proc. 14th IEEE Symp. on Computer Arithmetic*, pages 30–34, 1999.

[54] T. Han and D.A. Carlson. Fast area-efficient vlsi adders. In *Proc. 8th IEEE Symp. on Computer Arithmetic*, May 1987 1999.

[55] R.P. Brent and H.T. Kung. A regular layout for parallel adders. *IEEE Trans. Computers*, 31:260–264, 1982.

[56] A. Weinberger and J.L. Smith. A logic for high-speed addition. *Nat. Bur. Stand. Circ.*, 591:3–12, 1958.

[57] J. Sklansky. Conditional-sum addition logic. *IRE Trans.*, EC-9:226–231, June 1960.