Low Power Viterbi Decoder Designs

A thesis submitted to The University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2007

Wei Shao

School of Computer Science

CONTENTS

1. Introduction 1

1.1 Digital communication and coding 1

1.1.1 Information theory 3

1.1.2 Coding theory 6

1.2 Decoding Convolutional Codes 13

1.3 Channel Coding Applications 14

1.3.1 Channel coding application considerations 14

1.3.2 Low power applications 15

1.4 Objectives and summary of this work 16

1.5 Contributions of this work 16

1.6 Thesis Overview 18

2. Convolutional Coding and Viterbi Decoding Algorithm 19

2.1 Convolutional code structure 19

2.1.1 Tree and trellis representation of convolutional codes 22

2.1.2 Rate m/n codes 25

2.2 Distance properties of convolutional code 28

2.3 Viterbi decoding algorithm 31

2.3.1 Hard-decision Viterbi decoding algorithm 31

2.3.2 Soft-decision Viterbi decoding algorithm 34

2.4 Convolutional codes performance with Viterbi algorithm 37

2.4.1 Error event and union bounds 37

2.4.2 Convolutional code performance 39

2.5 summary 42

3. Viterbi decoder and its power dissipation 44

3.1 Viterbi decoder design 44

3.1.1 BMU design 45

3.1.2 PMU design 50

3.1.3 SMU design 52

3.2 Power dissipation in the Viterbi decoder 55

3.2.1 CMOS Circuitry power dissipation 56

3.2.2 Design flow and power estimation 58

3.2.3 Testing framework and noise generator design 60

3.2.4 Viterbi decoder power consumption 61

3.3 Summary 69

4. Low power adaptive Viterbi algorithm and decoder design 71

4.1 T-algorithms and adaptive T-algorithm 71

4.2 A new adaptive Viterbi algorithm 73

4.2.1 Error pattern in Viterbi algorithm 73

4.2.2 No-error code words path identification 75

4.2.3 A new adaptive Viterbi algorithm and decoder design 80

4.3 BER and power analysis of the proposed adaptive Viterbi decoder . . . 81

4.3.1 Matlab Simulation Results 82

4.3.2 FPGA simulation results 85

4.3.3 Estimated power consumption 86

4.3.4 Comparison of other low power designs 86

4.3.5 Possible applications 87

4.4 Summary 88

5. Low power SMU design 90
5.1 Design of SMU 90
5.1.1 Major SMU operations 91
5.1.2 Asynchronised SMU timing and design 93
5.2 New Trace Back SMU design 95
5.2.1 Timing feature of the trace back convergence 95
5.2.2 Overview of the new SMU architecture 96
5.2.3 64-bit global winner encoding 97
5.2.4 New trace back path architecture 98
5.2.5 Local Winner Memory 102
5.2.6 Global Winner Distributor 103
5.2.7 Output Generator 104
5.3 Timing in the new SMU design 107
5.3.1 Positive timing skew 107
5.3.2 Negative timing skew 109
5.3.3 Timing of the scaled new trace back design 112
5.4 Test results of the new SMU design 113
5.4.1 CMOS implementation results 113
5.4.2 FPGA implementation results 116
5.5 Summary 118

6. Low power PMU design 120
6.1 Existing low power implementations of PMU 120
6.1.1 Analog design 120
6.1.2 Low power compare-select-add (CSA) design 123
6.2 Low power design of the PMU 124
6.2.1 Performance analysis with BM and PM Capping 125
6.2.2 Low power implementation of the PMU with PM and BM capping 129

6.3 Test Results 131
6.4 Summary 132

7. Conclusion and Future Works 135
7.1 Conclusion 135
7.1.1 Summary of proposed low power Viterbi decoder designs 135
7.1.2 Contributions and new methods proposed for efficient decoding 140
7.2 Future works 142

Appendix 151

A. Matlab Programs 153
A.1 Matlab program for comparing union bound with simulated BER of Viterbi decoding 153
A.2 Matlab program for comparing union bounds of different code rates . . 155
A.3 Matlab program for comparing union bounds of different constraint lengths 157
A.4 Matlab program for comparing union bounds of hard and soft decision 159

B. The AWGN generator 160
B.1 The AWGN generator design overview 160
B.2 The f function implementation 162
B.3 The AWGN generator design verification 165

LIST OF FIGURES

1.1 Simple communication system. 4

2.1 A simple R=1/2, k=3 convolutional encoder 20

2.2 Code tree for R=1/2, k=3 convolutional code. 22

2.3 Trellis structure for R=1/2, k=3 convolutional code. 24

2.4 Encoder structure for R=2/3, k=4 convolutional code. 25

2.5 Trellis structure for R=2/3, k=4 convolutional code. 26

2.6 Trellis structure for R=2/3 punctured convolutional code with a mother code R=1/2, k=3. 27

2.7 Trellis structure for R=1/2, k=3 code with two highlighted code word paths begin and end in the same state. 28

2.8 State transition diagram of R=1/2, k=3 code. The input and output states are zero state. 29

2.9 The communication channel model using a convolutional code and ML decoding. 32

2.10 The trellis structure shows the Viterbi decoding process of a R=1/2, k=3 code. 33

2.11 The convergence of the code word paths in the Viterbi decoding process. 34

2.12 The normal probability distribution of a random variable. 36

2.13 The error event with Viterbi decoding. 37

2.14 Union bound and simulated BER for R=1/2, k=7 code with hard-decision Viterbi decoding and QPSK (Quadrature Phase Shift Keying) modulation. 39

2.15 Union bounds for R=1/3, 1/2, and 2/3, code with soft-decision Viterbi decoding and QPSK modulation. 40

2.16 Union bounds for R=1/2, k=3, 4, 5, 6, 7, and 8 code with soft-decision Viterbi decoding and QPSK modulation. 41

2.17 Union bounds for R=1/2, k=7 code with both Soft and Hard decision Viterbi decoding and QPSK modulation. 42

3.1 Classical three functional block of a rate 1/2 Viterbi decoder design. . . 44

3.2 The Euclidian distance between the hypothesised code word and the received signal. 47

3.3 Three-bit quantisation in a 2-dimensional space. 48

3.4 The butterfly state transition diagram represents state transitions of a convolutional encoder of constraint lenght k. 50

3.5 A rate 1/2, n states *PMU* architecture shows the recursively calculations of state metrics value. The Global Winner Generator, shown in dashed lines, provides global winners information for a trace back *PMU*. 51

3.6 A 4-state register exchange implementation of the *SMU* design. The bold arrows indicate the ideal path of the encoder states. 53

3.7 A possible trace back *SMU* implementation using memory. It also requires global winner information in order to reduce the trace back depth. 54

3.8 Design and verification process of the FPGA implementations 60

3.9 The test framework of the FPGA implementaion. 61

3.10 The optimum number of test bits for power simulations. 63

3.11 The Viterbi decoder power consumption at different Eb/No levels. . . . 64

3.12 Energy per corrected error of the Viterbi decoders of k=3 and 7. 65

3.13 The estimated dynamic power consumption of the Viterbi decoder with different constraint lengths at Eb/No=2dB. 66

3.14 Energy per corrected error of the Viterbi decoder with different constraint lengths at Eb/No=2dB. 67
3.15 The blocks dynamic power consumption of a standard (R=1/2, k=7) Viterbi decoder. 68

4.1 The error pattern in the code words and the corresponding decoded data of a R=1/2, k=7 Viterbi decoder at Eb/No=3dB. 74
4.2 Simple convolutional coded digital communication system model. . . . 76
4.3 Architecture to identify the zero Hamming distance path for a rate=1/2 and k=7 convolutional codes. 78
4.4 Architecture of the proposed 3-bit soft decision adaptive Viterbi decoder for rate=1/2 and k=7 code. 81
4.5 BER performance of the adaptive Viterbi algorithm with $L = 35$, R=1/2, k=7. 82
4.6 Comparison of pre-decoding and Viterbi decoding operations, R=1/2, k=7. 83
4.7 BER performance of adaptive Viterbi algorithm with L from 4 to 28, R=1/2, k=7. 84
4.8 Percentages of pre-decoding and Viterbi decoding operations, R=1/2, k=7. 85
4.9 BER performance from FPGA tests, R=1/2, k=7. 86
4.10 Estimated dynamic power consumption of the adaptive Viterbi decoder on Virtex4 XC4VSX35, R=1/2, k=7. 87

5.1 Memory architecture of the one-pointer trace back SMU. 92
5.2 The SMU architecture of the asynchronous design from [1]. 94
5.3 The trellis structure shows the Viterbi decoding process of a R=1/2, k=3 code. 95

5.4 The new R=1/2, k=7, 64-state *SMU* architecture, which consists of four major blocks. 96
5.5 Trace back path of the new *SMU* design. 98
5.6 The timing of the trace back path multiplexer. 100
5.7 The one stage trellis structure of the Trace back unit. 101
5.8 Local winner memory of the new *SMU* design. It uses latch registers to store local winner information. 102
5.9 Global Winner Distributor of the new *SMU* design. 103
5.10 Timing of the global winner buses A and B 104
5.11 Metastability and a simple flip-flop synchronizer. 105
5.12 Metastability simulation circuit in measuring τ. 106
5.13 Trace back status in positive timing skew situation. 108
5.14 Trace back gap caused by timing skew. 109
5.15 Trace back status at various time points. 110
5.16 Post-layout of the new SMU core. 114
5.17 BER performance from FPGA tests. 117

6.1 The conventional architecture of the ACS unit. 123
6.2 The low power ACS unit architecture proposed in [2]. 124
6.3 The BER performance variation of the Viterbi decoder with different BM and PM capping levels at Eb/No of 3dB. 125
6.4 The BER performance of the Viterbi decoder with the variation of BM capping level at Eb/No of 3dB. 126
6.5 The BER performance of the Viterbi decoder with the variation of PM capping level at Eb/No of 3dB. 128
6.6 The architecutre of the BM and PM capping ACS unit. 130
6.7 The BER performance of the proposed low power ACS design. 131
6.8 The power dissipation of a PMU with the new ACS architecture. 133

B.1 Architecture of the AWGN generator 161

B.2 Uniform segementation of $f(x)$ 162

B.3 A example of 8-bit non-uniform segementation of $f(x)$ 163

B.4 $f(x)$ architecture 164

B.5 Comparison of the uncoded BER 165

B.6 Comparison of the decoded BER 166

LIST OF TABLES

3.1 Branch weight scheme for 2-symbol, 3-bit quantisation. 50

3.2 Features of the Virtex-4 SX class devices. 60

5.1 Minimum and maximum trace back stages at a 0.18μm geometry. . . . 112

5.2 Minimum and maximum delays of each trace back stage for different geometries. 112

5.3 Minimum and maximum trace back stages at 50MHz and L=64. 113

5.4 Minimum and maximum trace back stages at 100MHz and L=62. . . . 113

5.5 Characteristics of the new SMU core 114

5.6 New SMU BER and power consumption in decoding 1/2 codes 115

5.7 New SMU BER and power consumption in decoding 2/3 codes 115

5.8 The proposed SMU power consumption comparing with other low power SMU designs at 1.8V and 180nm. 115

5.9 Number of slices the Viterbi decoder occupied with the new SMU architecture 116

5.10 Dynamic power consumption of the Viterbi decoder with the new SMU Implemented on FPGA 117

6.1 The optimum BM caps for R=1/2, k=7 Viterbi decoder. 127

6.2 The optimum PM caps for R=1/2, k=7 Viterbi decoder. 129

6.3 FPGA simulated power of a R=1/2 k=7 PMU with the new ACS architecture at 50MHz. 132

B.1 Non-uniform segment in 8-bit binary numbers. 164

THE UNIVERSITY OF MANCHESTER

ABSTRACT OF THESIS submitted by: **Wei Shao**

for the Degree of Ph.D. and entitled: **Low Power Viterbi Decoder Designs**

Date of submission: **30/03/2007**

This thesis represents the research work of developing new approaches for implementing Viterbi decoder designs to minimize computation complexity and power consumption. This work examines the decoding process of the Viterbi algorithm, the architecture of the Viterbi decoder, and the implementations of the basic functions. This enables the design problems to be discovered. Then a variety of low power design techniques are described and applied to the decoder design to improve its power efficiency.

The new designs are tested by simulations on both software and hardware. The results give a clear view of the improvement of the modifications and enable a novel general methodology for significantly reducing complexity of decoding convolutional codes to be proposed.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i Copyright in text of this thesis rests with the Author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the Author and lodged in the John Rylands University Library of Manchester. Details may be obtained from the librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the Author.

ii The ownership of any intellectual property rights which may be described in this thesis is vested in the University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

iii Further information on the conditions under which disclosures and exploitation may take place is available from the Head of the Department of Manchester Business School.

Acknowledgements

I wish to acknowledge Engineering and Physical Sciences Research Council (EPSRC), who kindly funded the project, and the previous works of low power Viterbi decoder designs accomplished by The University of Manchester, Queen's University of Belfast, and The University of Sheffield.

Many individuals have, in one way or another, influenced me during the work on this thesis - for this I am most grateful. In particular, I wish to thank Dr. Linda Brackenbury, who has guided me down the academic road, and without her, this thesis would not have been possible. I am truly fortunate to have a supervisor whom I can also consider to be a good friend and who has during the last couple of years continued to offer help and advice on my work as well as other aspects of life.

Amongst my colleagues in the Advanced Processor Technique (APT) group thanks are due to Jeff Pepper for providing technique support and to Eve, although she finished her PhD and left the school at the end of 2005, for advising design and testing tools and techniques. I am also grateful to the rest of APT group members, who took time from their busy schedules and so patiently listened to all my seminars.

Especially, I would like to give my special thanks to my parents and Qiuping, for their love and support during my engagement with this research.

DEDICATION

To the memory of my grandma
Yongjing Nie
who taught me the value of hard work

1. INTRODUCTION

"*Attention, the Universel! By kingdoms, right wheel!*" This prophetic phrase represents the first telegraph message on record. Samuel F. B. Morse sent it over a 16 km line in 1838. Thus a new era was born: the era of electrical communication.

Now, over a century and a half later, communication engineering has advanced to the point that earthbound TV viewers watch astronauts working in space. Telephone, radio, and TV are integral parts of our life. Long-distance circuits span the globe carrying text, data, voice, and images. Computers talk to computers via intercontinental networks. Wireless personal communication devices keep us connected wherever we go. Certainly great strides have been made since the days of Morse.

This thesis describes the research work that has been done in order to improve the power efficiency of the Viterbi decoder in digital communication systems. In particular, a novel adaptive Viterbi decoder is presented plus, a lower power *Path Metric Unit* (PMU) and *Survivor Memory Unit* (SMU) designs that have been developed. These are discussed together with the results from testing them.

1.1 Digital communication and coding

It is remarkable that the earliest form of electrical communication, namely telegraphy developed by Samuel Morse in 1837, was a digital communication system. Although Morse was responsible for the development of the first electrical digital communication system, the beginnings of what we now regard as modern digital communications system from the work of Nyquist in 1924. His studies led him to conclude that for binary data transmission (transmitting one of two numbers, 0 or 1) over a noiseless channel

of bandwidth W Hertz, the maximum pulse rate is $2W$ pulses per second without any cross symbol interference.

Hartley extended this work in 1928 to non-binary data transmission, while Kolmogorov and Wiener independently in 1939 and 1942, respectively, solved the problem of optimally estimating a signal in the presence of additive noise. In 1948 Claude Shannon established the mathematical foundation for information transmission and derived fundamental limits for digital communication systems. His work can arguably be considered as the true beginning of the information age.

Another important contribution to the field of digital communication is the work of Kotelnikov in 1947, who provided a coherent analysis and consequently a principle for optimal design of such systems. His work was later extended by Wozencraft and Jacobs in 1965, leading to the principles used to design the communication systems of today.

The work of Hamming in 1950 on error control coding to combat detrimental effects of channel noise completes the classic contributions to modern digital communication systems.

Of more modern contributions, the Viterbi decoding algorithm for trellis codes, proposed by Andrew Viterbi in 1967 is now found in almost all wireless communication systems. Efficient error control decoding makes mobile communication systems what they are today. The latest significant leap forward for improvements of communications systems was in 1993 with the discovery of the "turbo principle" by Berrou and Glavieux. The special turbo codes developed based on these principles can be efficiently decoded using a very powerful iterative signal processing approach. The resulting coding system performs very close to fundamental limits for a range of different channels. In practical terms, this leads to the most efficient use of bandwidth and power, which is very important for portable wireless devices.

In practice, the subject of digital communications involves the transmission of information in digital form from a source that generates the information to one or more

destinations. In particular, it includes the concepts of source coding including entropy and rate-distortion, the characterization of communication signals and systems, optimal receivers, carrier and symbol synchronization, channel capacity and coding, block and convolutional codes, signal design for band-limited channels, adaptive equalization, etc.

Theoretically, communication theory consists of two major domains: information theory and coding theory.

1.1.1 Information theory

Information theory is the study of how the amount of content in a stream of data may be evaluated, and how fast it may in principle be shipped from place to place by a given communication channel [3]. The channel may need the data in a specific form and may corrupt it by randomly introducing errors. The subject is thus built on discrete probability theory as its mathematical base. It is somewhat high level in hierarchy, giving bounds and existence proofs without always any explicit means of implementation.

When discussing information theory it is hard not to mention Claude Elwood Shannon (April 30, 1916 - February 24, 2001), an American electrical engineer and mathematician. He has been called "the father of information theory", and was the founder of practical digital circuit design theory. In 1948 Shannon published *A Mathematical Theory of Communication* in two parts in the July and October numbers of the *Bell System Technical Journal*. This work focused on the problem of how to best encode the information a sender wants to transmit. In this fundamental work he used tools in probability theory, developed by Norbert Wiener, which were in their nascent stages of being applied to communication theory at that time. Shannon developed information entropy as a measure for the uncertainty in a message while essentially inventing what is now known as the dominant form of "information theory".

One of the most fundamental results of this theory is Shannon's source coding

theorem, which establishes that on average the number of bits needed to represent a random variable X which is given by the entropy $H(X)$ and is defined as [4]:

$$H(X) = -\sum_{x \in X} p(x) log p(x) \tag{1.1}$$

where x and $p(x)$ represent the possible values of X and their probability, respectively. This equation plays a central role in information theory as measurements of information, choice and uncertainty, reflecting the real life fact that an unusual message contains more information than a normal one and thus may be more difficult for us to understand. Therefore, more bits are required in order to describe it more clearly than a normal message. Mackay [3] summarizes this theorem as: " *N independent identically-distributed (i.i.d.) random variables each with entropy H(X) can be compressed into more than NH(X) bits with negligible risk of information loss, as N tends to infinity; but conversely, if they are compressed into fewer than NH(X) bits it is virtually certain that information will be lost.*" When applying the source coding theorem to communications over a noisy channel, Shannon invented the noisy channel coding theorem. This states that reliable communication is possible over noisy channels provided that the rate of communication is below a certain threshold called the channel capacity. This is also called the Shannon limit or Shannon capacity [4].

Considering a simple communications process over a discrete channel, as shown in

Fig. 1.1: Simple communication system.

Figure 1.1, here X represents the space of all possible values of the messages received, and Y similarly is the space of all possible values of messages transmitted during a unit time over this channel. The possible rate of information transmission, R, would be obtained by subtracting the average rate of conditional entropy, $H_y(x)$, from the entropy of the source, $H(x)$ [3].

$$R = H(x) - H_y(x) \tag{1.2}$$

The conditional entropy $H_y(x)$ measures the average ambiguity of the received signal. The capacity C of a noisy channel is the maximum possible rate of transmission, i.e., the rate when the source is properly matched to the channel [3]:

$$C = MAX(H(x) - H_y(x)) \tag{1.3}$$

The theorem formally states that for a source entropy of H, *"if $H \leq C$ there exists a coding system such that the output of the source can be transmitted over the channel with an arbitrarily small frequency of errors (or an arbitrarily small equivocation). If $H > C$ it is possible to encode the source so that the equivocation is less than $H - C + \epsilon$ where ϵ is arbitrarily small. There is no method of encoding which gives an equivocation less than $H - C$"* [4].

This Shannon's information theory and his theorems have large impacts on the modern communication world:

1. They suggest a methodology to quantize a piece of information;

2. They describe the correlation between the uncertainty of information and its transmission speed;

3. They indicate the maximum transmission rate for a noisy channel at a certain noise level.

Nowadays, Shannon's limit becomes the ideal objective for most designers of the communication systems. Coding techniques are essential for a communication system to achieve such target.

1.1.2 Coding theory

Coding theory is more practical compared with other theories in the information theory domain. It is primarily concerned with finding the methods, called codes, for increasing the efficiency and accuracy of data communication over a noisy channel as close to the theoretical limit that Shannon proved as possible. These codes can be mainly subdivided into source coding (Entropy encoding) and channel coding (Error correction coding) [5]. A third class of codes are cryptographic ciphers [6], which implement concepts from coding theory and information theory in cryptography and cryptanalysis. This thesis is only concerned with channel coding, as it is widely used to improve the reliability of communication on digital channels y detecting and correcting errors [5].

Although there are many forms of coding schemes, they all have two basic features in common [7]. One is the use of *redundancy*. Coded digital messages always contain extra or redundant symbols. In fact, these "redundant" symbols are not really redundant as they contain the information to accentuate the uniqueness of each message so that the channel disturbance is unlikely to destroy the message by corrupting enough of the symbols in it. The second feature is *noise averaging* [7]. This is achieved by making the redundant symbols depend on a span of several information symbols. This means the redundant symbols not only make the sent message more distinctive but also contain the information of the transmitted message itself. Therefore, each symbol of the message actually contains less transmitted information and thus causes less damage when it is corrupted by noise.

Two kinds of codes are mainly used in modern communication: block codes and convolutional codes [5] [7]. This classification is based on the presence or absence of memory in the encoders for these two codes. An encoder for a block code is memoryless

as it maps a k-symbol input sequence into an n-symbol code words sequence. Therefore, each n-symbol output only depends upon a specific input k-symbol block and the encoder has no "memory" of other previous input symbols. For the block codes, there is no correlation between the encoded output code words. In contrast, the output of encoding a convolutional code is determined by the current input and a span v of the preceding input symbols. Each input is memorized by the encoder for a certain amount of the time span so that it affects not only the current output but also the next v output code words.

Although, codes can also be classified as linear or nonlinear, almost all the coding schemes used in practical applications are linear codes due to their significantly simplified mathematical representations. For this reason, the codes mentioned in this thesis are all linear unless otherwise specified.

1. **Block codes**. A block code is normally specified by values of parameters n, k, $R = k/n$, and d_{min}. These parameters indicate that the encoder encodes each k symbols input block into a n symbols output block. Therefore, the code has a rate of R equal to k/n. The minimum of *Hammming distance*, d, of the code is defined as d_{min}.

$$d_{min} = min(d) \tag{1.4}$$

The *Hammming distance* refers to the number of positions in which any two binary sequences differ from each other [7].

There are many types within linear block codes, like parity codes, Repetition codes [3], BCH (Bose, Ray-Chaudhuri, Hocquenghem) code [3], Hamming code [8], Reed Solomon codes [9], Reed Muller codes [5], or Perfect codes [10], etc. *Parity codes* $(n, n-1)$ were used in the early days. It was formed by using a single overall parity check bit after the information sequence. For instance, a

code word of a (4,3) parity code can be defined as a vector A, where

$$A = (a_1, a_2, a_3, a_1 + a_2 + a_3) \tag{1.5}$$

The first three symbols in the vector, a_1, a_3, and a_3, are the binary symbols containing information; the last symbol is the parity check symbol, which is the modulo-2 addition of the first three information symbols. As shown in this example, each code word in a block code can be divided into two portions. The first k symbols portion is always identical to the information sequence to be transmitted. Each of the $n - k$ symbols in the second portion is computed by taking a linear combination of a predetermined subset of information symbols. Parity checking is not very robust, since if the number of bits changed is even, the check bit will be valid and the error will not be detected. Moreover, parity does not indicate which bit contained the error, even when it can detect it. The data must be discarded entirely, and re-transmitted from scratch. On a noisy transmission medium a successful transmission could take a long time, or even never occur. Parity does have the advantage, however, that it is the best possible code that uses only a single bit.

In the 1940s Bell used a slightly more sophisticated code known as the *two-out-of-five code* [3]. This code ensured that every block of five bits (known as a 5-block) had exactly two 1s. The computer could tell if there was an error if in its input there were not exactly two 1s in each block. Two-of-five was still only able to detect single bits; if one bit flipped to a 1 and another to a 0 in the same block, the two-of-five rule remained true and the error would not be discovered.

Another code in use at the time was to repeat every data bit several times in order to ensure that it got through [3]. For instance, if the data bit to be sent was a 1, an $n = 3$ repetition code would send "111". If the three bits received were not identical, an error had occurred. If the channel noise is low, only one

bit will change out of three. Therefore, 001, 010, and 100 each correspond to a 0, while 110, 101, and 011 correspond to a 1, the majority identical bits within these three indicating what the original bit was. A code with this ability to reconstruct the original message in the presence of errors is known as an *error-correcting code* [10]. Although the repetition code is virtually able to correct and detect any number of errors, if the number of duplicate bits is large enough, it is extremely inefficient, as the throughput drops drastically when increasing the number of times each bit is duplicated in order to detect and correct more errors.

To identify errors, the transmitted bits just need to be arranged such that different incorrect bits produce different error results. Instead of repeating each transmitted data bit, the extra bits can be more efficient so that it needs less redundant bits.

During the 1940s Richard Hamming developed several encoding schemes that were dramatic improvements on existing codes; this is now known as Hamming Code [8] [3]. The key to his invention was to have the parity bits overlap, such that they managed to check each other as well as the data. This was a major milestone in coding theory, after which coding schemes become more complex and powerful than before. The ideal of "overlapping" also becomes the major principle of most coding schemes today [7]. A Hamming code is an (n, k) block code with $q \geq 3$ check symbols and

$$n = 2^q - 1 \qquad k = n - q \tag{1.6}$$

The code rate R is

$$R = \frac{k}{n} = 1 - \frac{q}{2^q - 1} \tag{1.7}$$

The minimum distance, d_{min}, of a Hamming code is independent of q and fixed

at

$$d_{min} = 3 \tag{1.8}$$

So it can be used for single-error correction or double-error detection [7].

Due to the "overlapping" feature, codes becomes more complex. Polynomials are thus introduced in representing these codes [7]. An (n, k) code word of the form

$$(a_0, a_1, \cdots, a_{n-1}) \tag{1.9}$$

can now be represented as a polynomial in x,

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \tag{1.10}$$

where $a_0, a_1, \cdots, a_{n-1}$ are the coefficients.

The *BCH* (Bose, Ray-Chaudhuri, Hocquenghem) *codes* are a generalization of Hamming codes which allow multiple error correction [3]. They were first discovered by A. Hocquenghem in 1959 and independently by R. C. Bose and D. K. Ray-Chaudhuri in 1960. A t-error-correcting BCH code has the following parameters:

$$n = 2^m - 1, \qquad n - k \leq mt, \qquad d_{min} \geq 2t + 1, \tag{1.11}$$

This BCH code is able to correct t and detect $2t$ errors.

Reed Solomon (RS) *codes* are a subset of BCH codes [9]. The code was invented in 1960 by I. S. Reed and Gustave Solomon. It works by first constructing a polynomial from the data symbols to be transmitted and then sending an over sampled plot of the polynomial instead of the original symbols themselves. Because of the redundant information contained in the over sampled data, it is possible to reconstruct the original polynomial and thus the data symbols even

in the face of transmission errors, up to a certain degree of error [7]. Today they are used in disk drives, CDs, telecommunication and digital broadcast protocols.

Before discovering RS codes, I. S. Reed also contributed in discovering the *Reed-Muller* (RM) *codes*, together with D. E. Muller. One of the important properties of RM codes is that they form an infinite family of codes, and larger RM codes can be constructed from smaller ones. RM codes were efficient and relatively easy to decode at the time, especially the first-order codes. In 1972, a RM code was used by Mariner 9 to transmit black and white photographs of Mars [11].

2. **Convolutional Codes**. Convolutional coding technique was first introduced by Elias in 1955 [12]. A binary convolutional encoder is normally represented by the values of three parameters: n, m, and k. The values of n and m indicates that each n-bit input yields a m-bit output so that the code has a rate of R, where

$$R = n/m \tag{1.12}$$

A convolutional encoder is considered as a finite-state machine. The parameter k is called constraint length which equals the shift register stages of the encoder. The principle of "overlap" is extensively used in convolutional codes. In a rate 1/2 convolutional code, for example, each input is "overlapped" with several previous inputs to produce each pair of encoded symbols. It is not able to divide coded sequence into blocks as each coded symbol pair is interlocked with its neighbours. This provides convolutional codes with great distance and error correction features. The Voyager program uses a convolutional code with a constraint length k of 7 and a rate R of 1/2 [13]. Longer constraint lengths produce more powerful codes, but the complexity of decoding operations increases exponentially with constraint length, limiting these more powerful codes to deep space missions. Mars Pathfinder, Mars Exploration Rover and the Cassini probe to Saturn use a k of 15 and a rate of 1/6; this code performs approximately 2 dB better than the

simpler k=7 code at an additional cost of 256 times in the decoding complexity.

3. **Turbo Codes**. Turbo codes are a new class of iterated short convolutional codes [14]. The method was introduced by Berrou, Glavieux, and Thitimajshima in their 1993 paper: "Near Shannon Limit error-correcting coding and decoding: Turbo-codes" [14]. Unlike convolutional codes, turbo codes can be systematic so that the message and parity bits are separate. The parity bits from the turbo code encoder are generated in different ways. For instance, a rate 1/3 turbo code encoder has two sets of parity bits. The first set is encoded with the original message sequence; however the second uses the sequence randomly permuted from the original message.

   Two *maximum a posterior* (MAP) decoders can be used to decode a 1/3 turbo code with the two sets of parity bits. The first decoder estimates the errors in the received message and corrects some of them. The output from the first decoder is then permuted to match the sequence used to encode the second set of parity bits. The new sequence together with the received second set of parity bits is used to identify errors. Because of this new sequence, some of the errors which cannot be corrected by the first decoder can now be corrected. After the second decoder, the received data has been actually decoded twice. This is referred to as one iteration of the turbo code decoding process. By repeating the described decoding process, the number of errors in the received data reduces every time. In fact, for the 1/2 rate turbo code in [14], a bit-error-rate (BER) of 10^{-5} at a signal-to-noise ($E_b/N_o = 0.7dB$) is achieved after 18 iterations. This result closely approaches the limit defined by Shannon's theorem. However, the main drawbacks of turbo codes are the relative high decoding complexity and high latency [14].

1.2 Decoding Convolutional Codes

Convolutional coding techniques show significant error protection features over block codes. The decoding techniques for convolutional codes, thus has become the subject of much research interest. There are three basic methods for decoding convolutional codes: *maximum-likelihood decoding of Viterbi algorithm* [15], *sequential decoding* [16], and *syndrome decoding* [7].

The Viterbi decoding algorithm was first introduced by A. J. Viterbi in 1971 [17]. By estimating the most-likely received sequence, the Viterbi algorithm achieves its optimum performance in BER. Since the most-likely received sequence is a relative measurement, then to achieve the most optimum performance, all possible code words need to be compared. This requires extensive hardware computation and storage. Much research work is therefore concerned with minimizing the computation complexity of the Viterbi decoding while increasing its performance.

Shortly after Elias discovered convolutional codes, Wozencraft [16] devised a decoding technique which is called "sequential decoding". It is a trial-and-error search decoding technique that provides performance that can meet or exceed that of Viterbi decoders. The Fano algorithm [18] and stack sequential decoding algorithm [19] are two major sequential decoding techniques. The main difference between the Viterbi algorithm and a sequential decoding algorithm is that the Viterbi algorithm is a "one-direction" decoding process which computes all paths of code words, whereas the sequential decoding only retains a minimum number of code words for a path and needs to "go back" if the path is not correct [18] [19]. Basically, a sequential decoder performs the search in a sequential manner always operating only on a single path. Each time the decoder moves forward a "trial" decision is made. If an incorrect decision is made, subsequent extensions of the path will be wrong. When the decoder recognizes this situation, it searches back and tries alternate paths until it finally decodes successfully. The drawback thus is the substantial amount of computation required to try alterna-

tive paths and recover the correct one. A major research topic, therefore, is to find the optimum parameters to allow quick recognition of an incorrect decision and quick recovery of the correct paths in order to minimize the computation problem [7].

Syndrome decoding sacrifices BER performance in exchange for simplified computation. One widely used syndrome decoding technique is *table look-up feedback decoding* [7]. Instead of estimating the correct data, a syndrome decoder seeks the errors in the received sequence. It calculates the syndrome which only contains the information of the error patterns in the received data and uses it against the pre-computed syndrome in a look-up table. Thus the corresponding error pattern can be identified and the correct data can then be recovered. A syndrome decoder using a look-up table requires simple hardware implementation. However, the drawback is the BER performance degradation. Another major type of syndrome decoding technique is called "*threshold decoding*". It was discovered by Massey [20] and can achieve relatively higher BER performance than *table look-up feedback decoding* while still requiring a simple implementation.

1.3 Channel Coding Applications

There are many applications using channel encoding. For example, a typical music CD uses a Reed-Solomon code to correct for scratches and dust. In this application the transmission channel is the CD read out system. Mobile phones also use powerful coding techniques to correct for the fading and noise of high frequency radio transmission. From data modems telephone transmission to NASA space programs, all of them employ powerful channel coding to combat noise.

1.3.1 Channel coding application considerations

The aim of channel coding is to find codes which transmit quickly, contain many valid code words and can correct or at least detect many errors. These aims are mutually

exclusive however, due to the redundancy and channel capacity correlation illustrated in Shannon's theorems. Therefore, different codes are optimal for different applications. The required properties of a code mainly depend on the probability of errors occurring during transmission. Therefore, to examine the source and properties of errors in a target implementation is essential for a coding application.

In a typical CD, the impairment is mainly dust or scratches and errors are mainly bursty [21]. Thus codes are used in an interleaved manner. For the channel with high continuous error probability, convolutional codes are widely used. Deep space communications are limited by the thermal noise of the receiver which is more of a continuous nature than a bursty nature. Concatenated RS/Viterbi-decoded convolutional coding were and are used on the Mars Pathfinder, Galileo, Mars Exploration Rover and Cassini missions to provide optimum BER performance [22]. The concatenated RS-Convolutional codes are also extensively implemented in standard satellite digital video broadcasting (DVB) systems [23]. Mobile phones are troubled by rapid fading. The high frequencies used can cause rapid fading of the signal even if the receiver is moved a few inches. Again convolutional codes are used to combat fading although it normally requires shorter constraint lengths. In the future, NASA missions will use Turbo Codes as standard to further enhance the quality of deep space communications [22].

For correcting continuous errors, block codes can also be used. The narrowband modems are limited by the noise present in the telephone network and is also modelled better as a continuous disturbance. Block codes are used instead of convolutional codes however, as it requires simpler implementations [7].

1.3.2 Low power applications

The most widely used technique for correcting errors in wireless systems is Viterbi decoded convolutional codes. In different forms, it is used in everything from V.3x-series modems, GSM, the voice channels of 3G and satellite DVB. As the market expands, more and more features, such as watching TV, receiving DVB etc., are being

put into handheld devices. The leading standard for mobile TV, DVB-H (Digital Video Broadcasting - Handhelds), has emerged from Nokia and been standardized by the European standards group ETSI, as EN 302,304, etc. This requires powerful error-correction codes be implemented. Viterbi decoder implementations are complex and dissipate a large amount of power. With the proliferation of battery powered mobile phones, power dissipation, along with speed and area, is a major concern in the decoder design. The requirement for lower power dissipation and smaller complexity has encouraged researchers to implement various power reduction techniques to decoder designs in order to improve their power efficiency.

1.4 Objectives and summary of this work

The main target of this work is to develop new Viterbi decoder designs for minimizing computation complexity and power consumption. This work examines the decoding process of the Viterbi algorithm, the architecture of the decoder, and the implementations of its basic functions. This enables the design problems, leading to inefficiencies and wasting the power consumption of the decoder, to be discovered. A variety of low power design techniques are described and applied to the decoder design in order to improve its power efficiency. The new designs are tested by simulations on both software and hardware. The results give a clear view of the improvement of the modifications and enable a novel general methodology for significantly reducing complexity of decoding convolutional codes to be proposed.

1.5 Contributions of this work

The following results have been achieved in this research:

1. By analyzing the Viterbi decoding process, the error-independence property of the Viterbi algorithm is identified as one of the major problems which affects

the power efficiency of the Viterbi decoder design at a high level. More specifically, the Viterbi decoding process is error-independent which means the decoding operation is applied on each one of the received code words without any consideration of their error probabilities. In the situation when a block of received data contains no error, the decoder power is wasted in trying to correct errors in the sequence. Therefore, to improve the power efficiency of a Viterbi decoder, a general methodology is proposed which transforms the decoder from error-independent to error-dependent.

2. To decode error-dependently means the decoder should run in an adaptive manner. There are some existing adaptive decoding methods for convolutional codes. Most of the adaptability is achieved by approximating the calculation of the likelihood measurement. In this work, a new adaptive algorithm is proposed which can detect the sequence which has no error prior to the decoding. Thus the Viterbi decoding operation can be avoided to save power. This adaptive technique has been implemented on a FPGA and demonstrates a significant power saving at low noise levels.

3. In a Viterbi decoder, the *Survivor Memory Unit* (SMU) is a vital part of the design. So far, classical implementations of the SMU employ the register exchange or the trace back approaches. In the conventional trace back implementation, a read-write RAM architecture is generally adopted. However, it suffers from complex control circuits and speed penalty. In this research, a new approach to implement the trace back algorithm targeted at low power applications is proposed. The SMU design based on this new architecture is a mixed synchronous and asynchronous circuit. However, it has no handshake overhead compared to most asynchronous architectures. Post-layout simulation results on a $.18\mu m$ process show the new architecture saves more than 84% of the power dissipated compared with a SMU design using a low power logic family.

4. In the Viterbi decoder, the *Path Metric Unit* (PMU) consists of different function units. Modifications, such as capping the branch and path metrics, are proposed to improve the decoder power efficiency at the logical level.

1.6 Thesis Overview

In the next chapter, the convolutional code structure is introduced with a detailed discussion of the distance properties and BER performance of convolutional codes. Chapter three focuses on the principles of the Viterbi decoding algorithm and the decoding process is represented using a Markov model. Basic concepts related to the Viterbi algorithm, such as hard/soft-decision decoding, punctured codes, etc., are also introduced in this chapter. From chapter four to chapter seven, various low power Viterbi decoder implementations are discussed. Chapter four describes the classical 3-block Viterbi decoder architecture and several existing lower power designs with a power analysis. Chapter five discusses the proposed adaptive Viterbi decoding algorithm and its implementation on a FPGA, while chapter six proposes a mixed synchronous/asynchronous SMU design. In chapter seven, the low power modifications on the PMU of the Viterbi decoder are discussed and analyzed with power simulations. Finally, chapter eight gives some conclusions and indicates the directions that further interesting work could take.

2. CONVOLUTIONAL CODING AND VITERBI DECODING ALGORITHM

The development of convolutional codes is quite different from block codes. For block codes, algebraic properties are very important criteria in constructing codes with good error protection performance. This is not the case with convolutional codes. Most convolutional codes with good error protection performance have been found by computerized searches of large numbers of codes to locate those with good distance properties [7]. This chapter studies the detailed structure of convolutional codes and provides significant insight into how the code properties influence BER performance when using the Viterbi decoding algorithm.

2.1 Convolutional code structure

A convolutional encoder with constraint length k consists of a k-stage shift register. A simple $k = 3, R = 1/2$ convolutional encoder is shown in Figure 2.1. Information symbols are shifted in at the left and the two modulo-2 adders yield two coded symbols which form a single code word. The connections between the shift register and the modulo-2 adder can be represented by the coefficient of polynomials. The upper and lower connections illustrated in Figure 2.1, for instance, can be described by the polynomials $g_1(x) = 1 + x^2$ and $g_2(x) = 1 + x + x^2$, respectively. Similarly, the input can also be represented as a polynomial $I(x) = i_0 + i_1x + i_2x^2 + \cdots + i_jx^j + \cdots$ where the coefficient i_j is the binary information symbol at time j. With this representation, the outputs of the convolutional encoder can be described as the multiplication of the input polynomial $I(x)$ and the connection polynomials. The upper and lower outputs

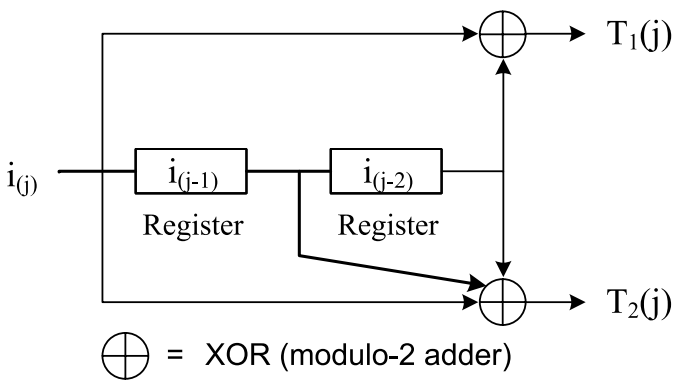


Fig. 2.1: A simple R=1/2, k=3 convolutional encoder

in Figure 2.1, can thus be represented as $T_1(x) = I(x)g_1(x)$ and $T_2(x) = I(x)g_2(x)$.

Another way of representing the encoder is to use a generator matrix, G, since the output of a convolutional encoder can also be thought of as the convolution of the impulse response of the encoder with the input sequence [7]. For example, the encoder in Figure 2.1 will produce a sequence of 11 01 11 00 00 ... if a single one followed by zeros is shifted into it. This sequence is thus the impulse response of a single one for this encoder, and the generator matrix, G, can be constructed as

$$G = \begin{pmatrix} 11 & 01 & 11 & 00 & 00 & 00 & \cdots \\ 00 & 11 & 01 & 11 & 00 & 00 & \cdots \\ 00 & 00 & 11 & 01 & 11 & 00 & \cdots \\ & & & \vdots & & & \end{pmatrix} \tag{2.1}$$

where each row of this matrix is the right-shifted version of the impulse response of a single one. Thus, the encoded sequence, Y, can be represented by multiplying the input vector X with G

$$Y = XG \tag{2.2}$$

and the output sequence can be produced by modulo-2 adding the rows corresponding

to the 1s in the input sequence. For instance, the output sequence corresponding to the input sequence $X = 1\ 0\ 1\ 0\ 0\ldots$ is obtained by adding rows 1 and 3 of G to give $Y = 11\ 01\ 00\ 01\ 11\ 00\ldots$.

The convolutional codes can be either non-systematic or systematic depending on the generator polynomial [7]. For example, if the generator polynomials $g_1(x) = 1$ or $g_2(x) = 1$ for a rate 1/2 code, the information sequence would appear directly in the output and the code becomes systematic. One of the advantages of a systematic code is that it is simple to extract the information sequence for a decoder. The decoder only needs to identify the error positions and flip the corresponding bits of the information sequence within the received data. If the decoder estimates no error, the information sequence can be used directly as the output. This may potentially save significant decoding computations. However, the drawback for systematic codes is the reduced noise-averaging feature. Since the original information sequence has no redundancy, it has no protection over noise. Therefore, the information sequence contained in the code words sequence will not help reduce any effect of the noise and the code is less effective in correcting errors.

Conversely, a non-systematic convolutional code does not directly contain the information sequence and it is much harder for the decoder to estimate the information digits without employing a complex decoding process. Wozencraft and Reiffen [24] have shown that for any non-systematic code there is a systematic code with precisely the same set of initial code words with the same minimum distance d_m. This result indicates that there is no advantage in using non-systematic codes where the decoder makes the decoding decision based on the initial code word only, e.g. threshold decoding for instance. However, for non-systematic codes with a sequential or maximum likelihood decoder which examines received digits well beyond the initial code word before making the decoding decision, then non-systematic codes show an inherent superiority in BER performance over systematic codes [24].

Another feature of a convolutional code is the random length of the code words.

The generator matrix shown in equation 2.1 is a semi-infinite matrix [7]. Thus, the length of a code word depends on the length of the information sequence which may be infinite. Tree and trellis diagram can be used to more clearly describe the relationship between input and output sequences of infinite length.

2.1.1 Tree and trellis representation of convolutional codes

There are two major methods to describe how the convolutional code word progresses with the possible input sequence. One of them is the code tree diagram as shown in Figure 2.2. The diagram in Figure 2.2 illustrates the code words structure for the

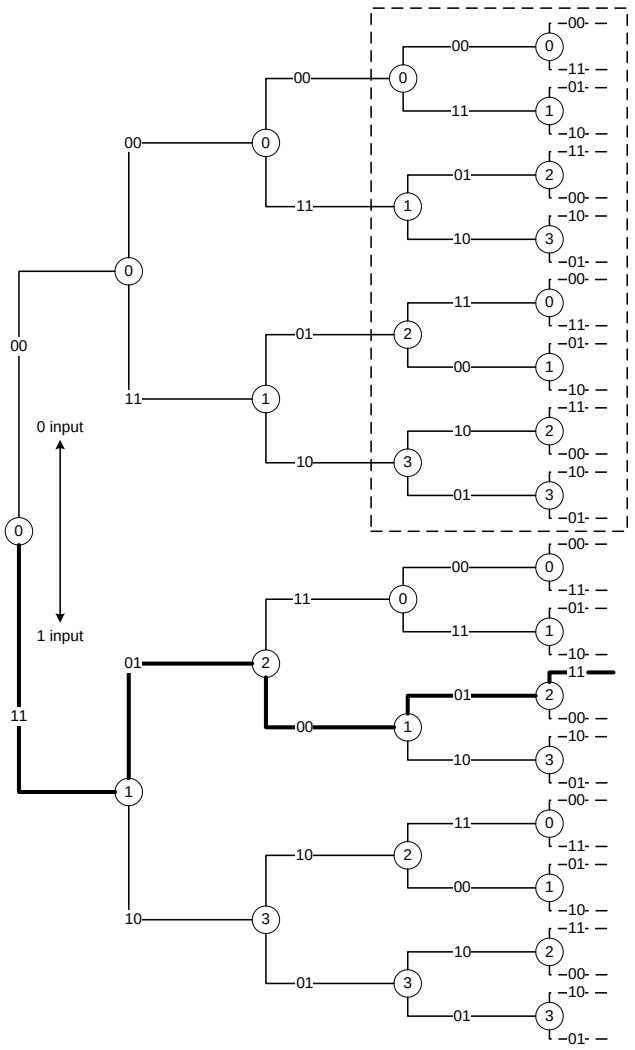


Fig. 2.2: Code tree for R=1/2, k=3 convolutional code.

convolutional code described in Figure 2.1. In this diagram, each branch of the tree represents a single input symbol where the upper branch corresponds to an input 0 and the lower branch corresponds to an input 1; the nodes in Figure 2.2 indicate different encoder states. Therefore, any input sequence can be traced through a path in this diagram which forms the corresponding code word. This path can also be called the code word path. For instance, a 1 0 1 0 0 input sequence representing the highlighted path in Figure 2.2 and gives an 11 01 00 01 11 output code word path. By observing this code tree diagram, it is clear that the number of possible code word paths grow exponentially as the length of the input sequence increases. This is the source of one major difficulty in decoding a convolutional code.

Although the code word paths may grow endlessly, they are not always different from each other. In Figure 2.2, for example, at the 4th step, the upper half of this code tree, which is marked as a dashed block, is identical to the lower half of the code tree. For the code in Figure 2.2, which has a constraint length of 3, each input symbol affects the output over 3 time steps. Therefore, the output becomes identical after three steps if the following input symbols are the same. This fact indicates one of the *merging* features of the convolution code and is the key to the Viterbi algorithm which will be discussed later. Because of the merger of code word paths, the code tree can also be represented in an alternative trellis structure with branches connecting a limited number of states. Figure 2.3 shows the trellis for the convolutional code represented in Figure 2.2. This trellis diagram is basically a state transition diagram over time steps. For this code, there are four possible encoder states. The convention is that each row of nodes representing the same state of the encoder at different time steps. At every time step there are two branches output from each node where an input 0 to the encoder corresponds to the upper branch and a 1 input to the lower branch. As in the code tree, the input sequence 1 0 1 0 0 corresponds to the particular path, shown with thick arrows, through the trellis.

A trellis diagram provides the best method of describing the Viterbi decoding al-

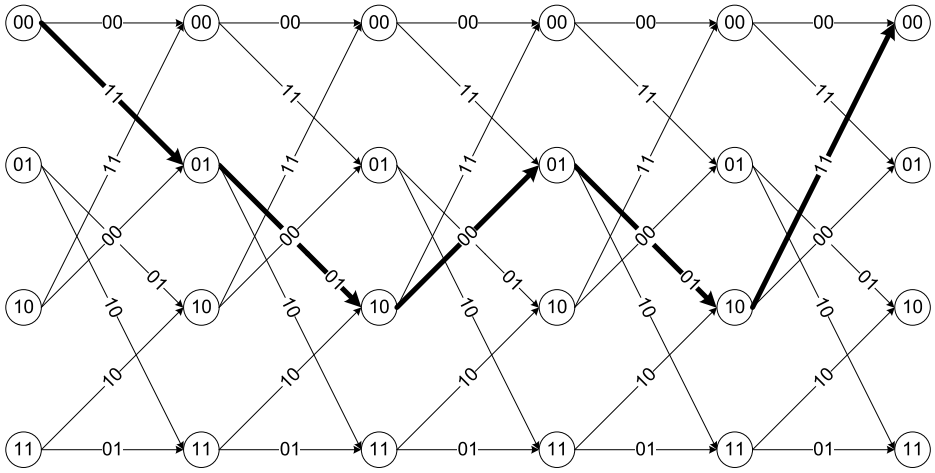


Fig. 2.3: Trellis structure for R=1/2, k=3 convolutional code.

gorithm. The significance of the trellis diagram is that the number of nodes in the trellis does not continue to grow as the length of the input sequence increases, because the redundant portions of the code tree have been merged. Ideally, the best decoding method for convolutional codes is to identify a code word path through the trellis which minimises the actual number of information symbol errors [7]. However, for the decoder to do so results in hardware that is hard to implement. More practically, it is easier to choose the code word path which matches the received sequence as closely as possible. The problem is that this procedure does not guarantee that by making the code word error rate small it will also minimise the bit error rate for the information sequence. In order to achieve the optimum BER performance, a convolutional code should also has the property that the minimum number of errors in the code word path also results in the minimum error number in the decoded information sequence. These important properties can be referred to as the distance property of the convolutional code and will be discussed later.

2.1.2 Rate m/n codes

One of the major parameters for a convolutional code is the code rate R. In the previous examples, a rate 1/2 code has been discussed. Code rates other than 1/2, such as 1/3, 2/3 or 7/8 etc., are also used in many applications. Figure 2.4 shows the encoder architecture for a rate 2/3, $k = 4$ convolutional code. Here two information

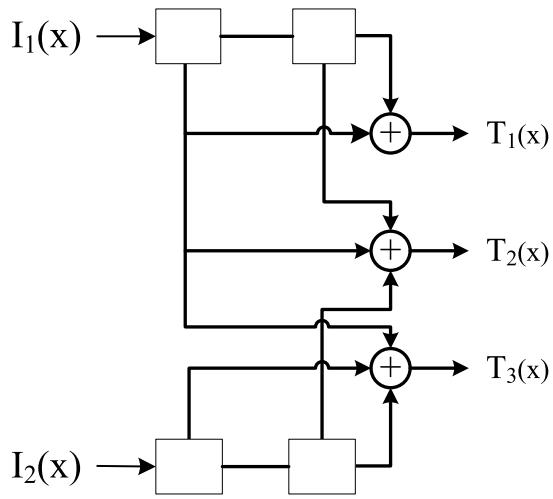


Fig. 2.4: Encoder structure for R=2/3, k=4 convolutional code.

symbols are shifted into two register lines at the same time, and three encoded symbols are generated by the modulo-2 addition of current and previous input of these two channels. The generator polynomial for this example can be defined by the matrix [7]

$$G(x) = \begin{pmatrix} 1+x & 1+x & 1 \\ 0 & x & 1+x \end{pmatrix} \tag{2.3}$$

Thus, the output vector is given by

$$[T_1(x)\ T_2(x)\ T_3(x)] = [I_1(x)\ I_2(x)]\, G(x) \tag{2.4}$$

The trellis structure for this code is shown in Figure 2.5. Compared with the $R = 1/2$ trellis in Figure 2.3, there are still four states since the number of shifter register in the encoder is the same. However, there are four possible output branches from each

node to represent the four possible combinations of every two input symbols.

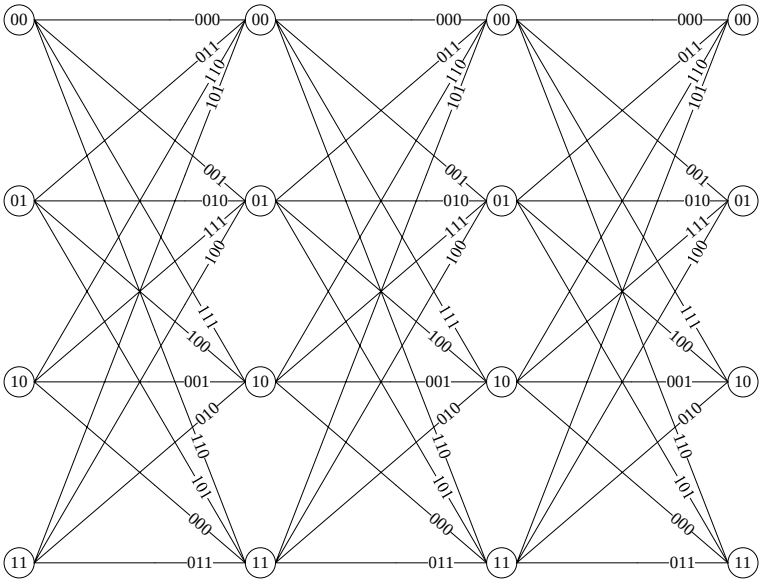


Fig. 2.5: Trellis structure for R=2/3, k=4 convolutional code.

In a generalised form, a $R = m/n$ convolutional code with the constraint length of k can be described in a trellis with 2^{k-m} states where 2^m branches come out of each state. The key advantage of a convolutional code with a high code rate is the increased data rate for transmission. For example, the information transmission rate for a convolutional code of $R = 7/8$ is more than 2.5 times higher than a $R = 1/3$ convolutional code. The major drawback of a high rate convolutional code is the reduced deocding accuracy and increased decoding complexity. The branch number for each node in the trellis equals 2^m which grows exponentially with an increase of m. This makes the implementation of the decoding algorithm much more difficult.

For example, a rate 7/8 code has 128 branches out of every node. To decode this code, the decoder has to choose one out of 128 branches compared with only one out of 2 for a rate 1/2 code. This is a fairly serious implementation difficulty. To avoid this problem, m needs to be minimised. In practice, a $R = 1/n$ code is normally converted into a $R = m/n$ code by deleting $(m-1)n$ symbols from every mn encoded symbols. This produces a $R = m/n$ code. For instance, suppose every fourth encoder output is

deleted from the $R = 1/2$, $k = 3$ code discussed before. This gives a new code of a rate 2/3 and is often refers to as a $R = 2/3$ *punctured* code of a mother code $R = 1/2$, $k = 3$. This code may still be decoded as a 2/3 code with an implementation difficulty as described before; alternatively, as it is generated from a $R = 1/2$ code, it can also be decoded as a $R = 1/2$ code with a dummy symbol within every other encoded symbol pair. The equivalent $R = 1/2$ trellis structure of this punctured code is shown in Figure 2.6. This trellis is the same as the $R = 1/2$ trellis in Figure 2.3 except one

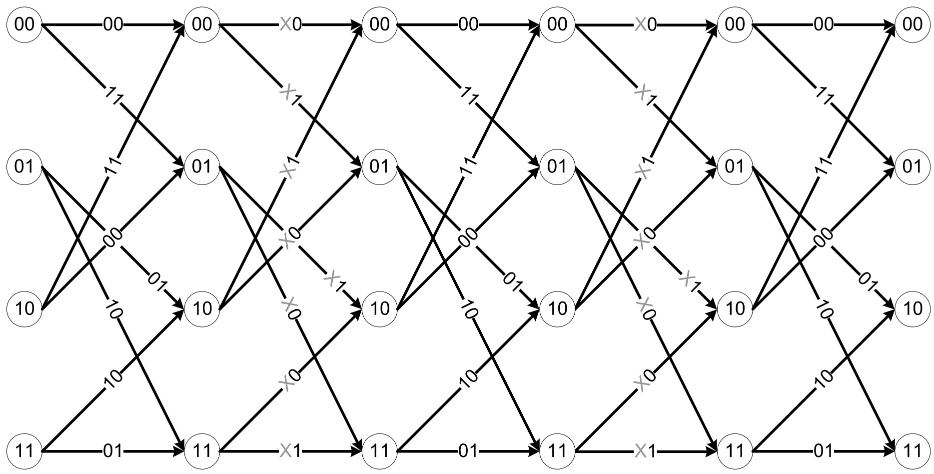


Fig. 2.6: Trellis structure for R=2/3 punctured convolutional code with a mother code R=1/2, k=3.

symbol is missing in every other branch and indicated by a X. Punctured codes have been shown to offer nearly equivalent performance compared to the best code with the same rate and thus are to be preferred in most cases because of their simplicity [25].

For a convolutional code, not only the code rate affects the capability of error protection; the type of decoding algorithm, the constraint length, the generator polynomials, the specific hardware configuration, or even the error characteristics may all contribute to the BER performance variation in a coding system design. It is, thus, impossible to give an absolutely accurate error rate of a particular convolutional code. However, by analysing the distance properties of the convolutional code a good estimation can be

achieved [7].

2.2 Distance properties of convolutional code

For a block code, the distance property which determines the performance of the coding is the minimum Hamming distance, d_{min}, between two finite code words. The concept of d_{min} for a convolutional code is similar. Suppose two different code word paths of a $R = 1/2$, $k = 3$ code are generated by two different input information sequences. Here the term "two different code word paths" refers to two different code word paths which start and terminate at the same state and have lengths equal or larger than a minimum length, $L_{min} = k$. To produce these two different paths at least one symbol in the input sequences is different. Once a different input symbol is shifted into the encoder, it affects the encoded output for 3 time steps, and thus gives a minimum length $L_{min} = 3$ for any two different code word paths. An example of two code word paths, p_0 and p_1, with $L_{min} = 3$ are shown in the trellis diagram in Figure 2.7. Since p_0

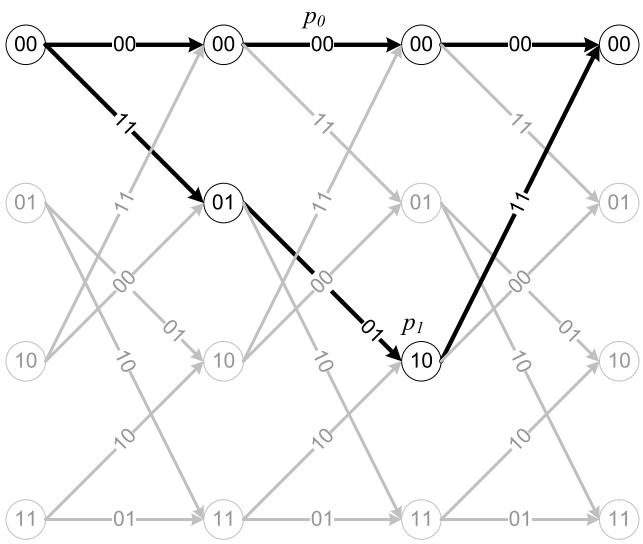


Fig. 2.7: Trellis structure for R=1/2, k=3 code with two highlighted code word paths begin and end in the same state.

is an all-zero path, the Hamming distance, $d = 5$, between p_0 and p_1 equals the weight of p_1. By observing this trellis, one can tell that $d = 5$ is also the minimum Hamming

distance between the all-zero path and any other path which begins and ends in the zero state. Since the code considered is a linear code, this result can be applied to any two code word paths without loss of generality. It can be concluded that this $R = 1/2$, $k = 3$ code has a minimum Hamming distance, $d_{min} = 5$. This d_{min} is also called the *free distance* and formally defined in [7] as the weight of the minimum-weight path which begins and ends in state zero.

To get the full set of the distance structure of a convolutional code is more difficult. One method described in [7] is to use the generating function for computing the gain of the state flow diagram. Figure 2.8 shows an example of the state transition diagram for the $R = 1/2$, $k = 3$ code. Every node in the diagram indicates a state of the encoder

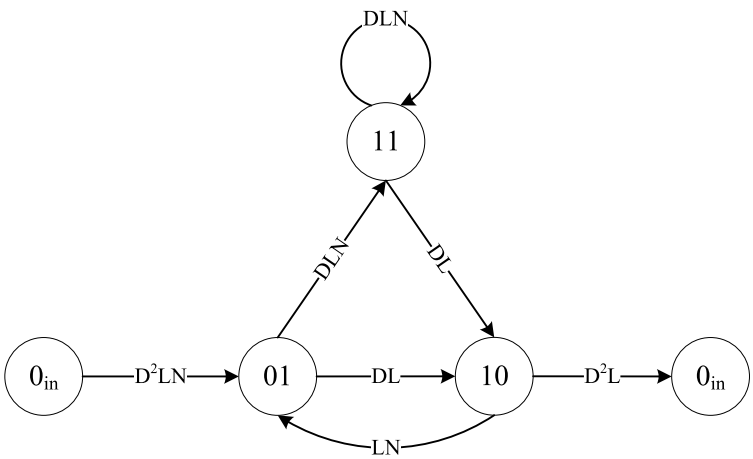


Fig. 2.8: State transition diagram of R=1/2, k=3 code. The input and output states are zero state.

where the input and output nodes are both zero state. This diagram is a modified version of the normal state transition diagram and is more appropriate to indicate the state transitions for the path beginning and ending in zero state. In the diagram, the gain of each branch is given by the product of $LN^{W_i}D^{W_o}$ [7], where L, N, and D are the indeterminates which represent the length of the input sequence, the weight of the input sequence and the weight of the output sequence, respectively; W_i and W_o are the values of the input and output weight for the particular branch. The generating function for this diagram is given by [7] as

$$
\begin{aligned}
T(D,L,N) &= \frac{D^5L^3N}{1-DL(1+L)N} \\
&= D^5L^3N + D^6L^4(1+L)N^2 \\
&\quad +D^7L^5(1+L)^2N^3 + \cdots \\
&\quad +D^{5+k}L^{3+k}(1+L)^kN^{1+k} + \cdots
\end{aligned} \tag{2.5}
$$

This shows the $R = 1/2$, $k = 3$ code provides one path of input weight 1, output weight 5 with a length of 3; two paths of input weight 2, output weight 6 with lengths 4 and 5; and so on.

For a convolutional code, the weight structure is determined by three parameters, the code rate R, constraint length k, and generator polynomials $g(x)$. On the basis of the weight structure, the performance of a convolutional code with a specific parameter set can be estimated. To achieve good error protection performance, the parameter set of a code should be carefully chosen according to two major distance criteria. First, a good code should have the free distance as large as possible. This makes each code word path more distinctive and more difficult for the decoder to make a wrong choice. Secondly, on the other hand, the distance between the input sequences, which produce two minimum distance paths, should be kept as small as possible. Thus, even when the decoder chooses a wrong path, the number of errors in the decoded data can still be minimised. The second criterion is very important as the performance of a code could be changed completely if this is not carefully complied with. One extreme case is the so-called *catastrophic error-propagating* codes [26]. In the occurrence of catastrophic error-propagation, a small number of errors in the received code word can result in choosing another code word for which the corresponding input sequence yields infinite number of errors.

In terms of generator polynomials, Massey and Sain [26] have obtained the conditions resulting in such catastrophic codes. In terms of the encoder state diagram, this

implies a state transition yielding a zero weight codeword (other than the self-loop of zero weight around the state S_0). For instance, consider the $R = 1/2$, $k = 3$ code with generator polynomials $g_1 = (1+x)$ and $g_2 = (1+x^2)$. Here an all-zeros input sequence produces the output sequence 00 00 00 ... while an all-ones input sequence gives the output sequence 11 10 00 00 00.... Thus, any two errors in the first three output symbols will result in choosing the all-zeros sequence instead of the all-ones sequence or choosing the all-ones sequence instead of the all-zeros sequence. This causes a major failure in decoding and gives an estimated information sequence with an infinite number of errors.

2.3 Viterbi decoding algorithm

The Viterbi decoding algorithm is a most-likelihood (ML) decoding algorithm for convolutional codes. A convolutional encoding is a discrete-time Markov process in which the sequence of the encoder states can also be seen as a Hidden Markov Model. These states are "hidden" and only the received code word sequences from the encoder are observable to the decoder. Thus, the Viterbi algorithm decodes the transmitted information by estimating the encoder states based on the received code word sequences.

2.3.1 Hard-decision Viterbi decoding algorithm

Considering the communication channel model in Figure 2.9, it describes the major concept of ML decoding for a convolutional code. As this figure shows, a binary information sequence $\{X_t\}$ is encoded into a code word sequence $\{Y_t\}$, which forms a path p of length L, and sent to a discrete noisy channel. The channel noise cause errors $\{E_t\}$ to be added to the transmitted data and gives the received sequence $\{R_t\}$ which forms a different code word path p' of length L. To recover the transmitted information sequence, the ML decoder performs two stages of operations. At the first stage, the decoder compares the p' with all possible code word paths by computing the

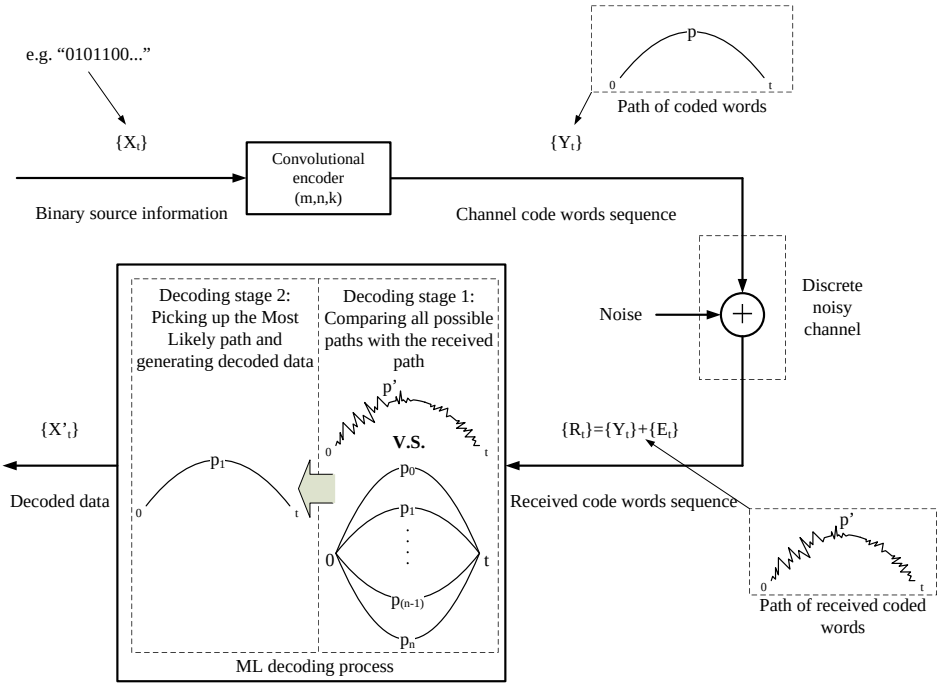


Fig. 2.9: The communication channel model using a convolutional code and ML decoding.

Hamming distances between every one of these paths. These Hamming distances are called *path metrics* (PM) and provide the measurements of the likelihoods for these paths over the length L. Based on the likelihood information, at the second stage the decoder chooses the most-likely path indicated by the smallest path metric. Thus, an estimated output $\{X'_t\}$ can be generated from the selected path. Channel noise may cause errors in p' and gives a Hamming distance d_e between p' and transmitted path p. However, as long as the Hamming distance between p' and p is smaller than all other possible code word paths, the p will always be chosen. Therefore, errors can be corrected in the decoded data.

This ML decoding approach seems extremely difficult to implement since the number of possible paths for comparing grows exponentially with the path length. By investigating the trellis, one may discover the fact that although at first the number of paths does grow with the length of an input sequence; however, because of merging, it becomes possible to discard a number of paths at every node so that exactly balances the number of new paths that are created. Thus, it is possible to maintain a relatively

small number of paths that always contain the ML path. In 1967 Viterbi invented a simple iterative process to implement this approach which is known as the Viterbi algorithm [17].

The trellis in Figure 2.10 shows the Viterbi decoding process for the $R = 1/2$, $k = 3$ code. For the input symbols sequence 0 0 1 1 0 0 0 0 eight pairs of encoded symbols,

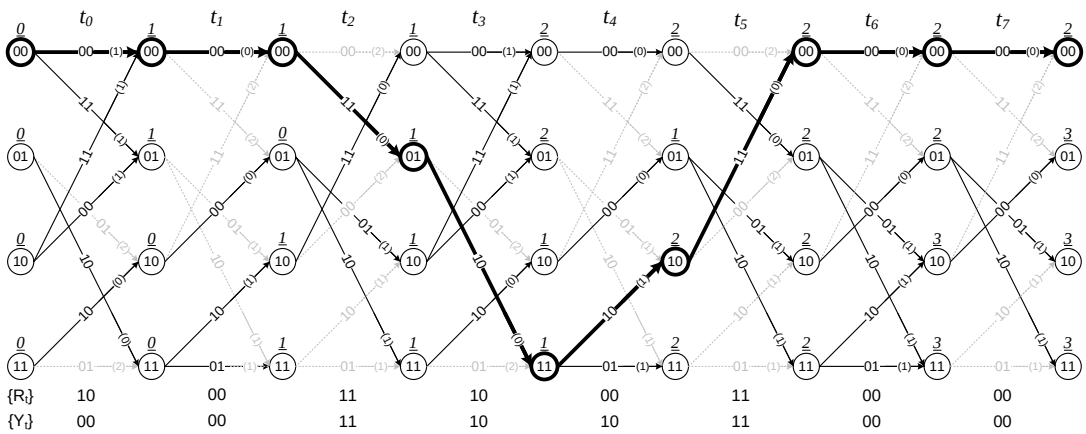


Fig. 2.10: The trellis structure shows the Viterbi decoding process of a R=1/2, k=3 code.

00 00 11 10 10 11 00 00 are generated. Suppose errors occur at time t_0 and t_4. This gives an incorrect code word sequence 10 00 11 10 00 11 00 00. At each time step, the Viterbi algorithm computes the Hamming distance between the received symbol pair and the expected symbol pair for every branch. This gives the weight of this branch and is normally called the *branch metric* (BM). The BM of each branch is shown as the number in the bracket. Although each state has two input branches, the Viterbi algorithm (VA) only allows one which is selected as the survivor branch, indicated as solid lines in the trellis. For each state, the VA makes a selection by adding the BM to the PM from last time step and chooses the branch having the smaller accumulated PM value. The PM of the survivor then becomes the PM of that state for the calculation at the next time step. The accumulated PMs are shown as the underlined numbers in this trellis. These operations for each time step can be summarised as an add-compare-select (ACS) process which is the key of the Viterbi algorithm. This process is continued each time a new branch is received so that the PMs of all survivor paths at

time t_7 are obtained. In order to form the output, the path with the smallest PM of 2, which is highlighted in this figure, is traced back through the trellis. The information sequence can thus be recovered without errors.

In fact, to produce the decoded information symbols it is not necessary to always identify the most-likely path. Considering the trellis in Figure 2.11, which is the ex-

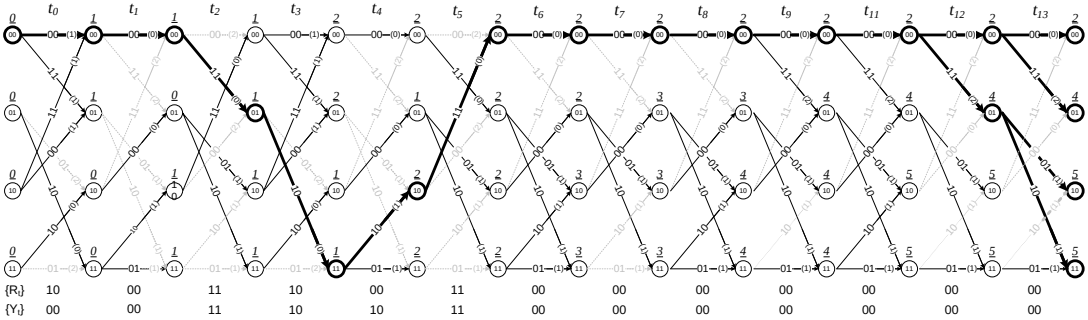


Fig. 2.11: The convergence of the code word paths in the Viterbi decoding process.

tended trellis of Figure 2.10, it shows all the survivor paths at the end of the trellis merge into a single path over the first 12 time steps. This is the result of ACS operations over a large enough time span so that the branches propagated from the most-likely path terminate all other paths. In this case, there is no need to identify the most-likely path at the end of the trellis. The decoded output can be generated from the exclusive merged path of the first 12 time steps. However, the overhead of this approach is the increased length of the paths. It obviously requires extensive storage for the path information compared with previous approach. This tends to be an important design decision which affects the complexity of an implementation and will be discussed in the next chapter.

2.3.2 Soft-decision Viterbi decoding algorithm

The likelihood measurement of the decoding process described before uses the Hamming weight or the Hamming distance. In these measurements, the difference between two symbols is quantized into a binary level and can be represented using a one bit binary

number. This distance quantization method is in fact based on the assumption that there is no difference between the errors caused by channel noise so that all errors can be represented by the same level. However, this assumption is too simple to reflect real communication channels.

In practice, the digital signal is still transmitted by analogue waveforms. Although the noise applied to the transmission waveforms takes many forms, they are all analogue in manner one way or the other. In communications, the additive white Gaussian noise (AWGN) channel model is one in which the only impairment is the linear addition of wideband or white noise with a constant spectral density (expressed as watts per hertz of bandwidth) and a Gaussian distribution of amplitude. The model does not account for the phenomena of frequency selectivity, interference, nonlinearity or dispersion. However, it produces simple but more realistic mathematical models for many studies and simulations of communication systems. In the AWGN channel, the noise signal as a random variable x has a probability density function of the (Gaussian) normal distribution with mean μ and variance σ^2,

$$P(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\ \mathtt{exp}\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{2.6}$$

In this function, the variance σ^2 is determined by the noise levels. Figure 2.12 shows this probability function $P(x)$ with the parameter set as $\mu = 0$ and $\sigma = 1$, where x is the noise variation and $p(x)$ is its probability. This is also called the standard normal probability distribution function and has the form

$$P(x) = \frac{1}{\sqrt{2\pi}}\ \mathtt{exp}\left(-\frac{x^2}{2}\right). \tag{2.7}$$

The plotted Figure 2.12 of the normal probability distribution function indicates the fact that the strength of the random noise signal reduces as the probability grows, with the highest probability when there is no noise at all. Therefore, for the communication channel with AWGN, the closer to no noise the signal is the higher the probability of

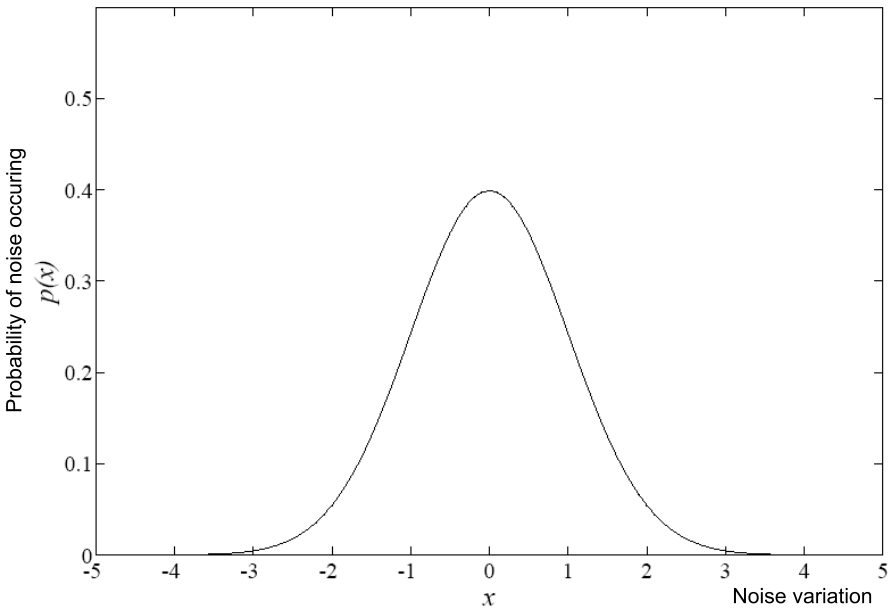


Fig. 2.12: The normal probability distribution of a random variable.

this occurring; in other words, the stronger the signal received the more likely it is to be correct. This fact indicates that the level of a received signal also provide the likelihood information. Therefore, each received signal is often quantized to N_q number of regions where N_q is the number of quantization levels. This is called *soft-decision* coding. The more the quantization levels of a signal, the more information of the signal likelihood is given to the decoder and the more accurate the decision the decoder can make. From an implementation point of view, however, it is desirable to make the number of quantization levels relatively small. This minimizes the complexity of the analogue-to-digital converter and also the number of bits involved in computing the metrics in the Viterbi algorithm. In many applications, eight-level quantization is used as a standard.

The soft-decision Viterbi algorithm is a simple modification of the hard-decision decoding process. The Hamming distance represented BM is replaced by the soft-decision weight represented with more binary bits. All other decoding operations remain the same. For an appropriate number of quantization levels, the increase of implemen-

tation complexity is not significantly different from that of a hard-decision decoder. This is a major advantage of the Viterbi algorithm [17]. The most significant factor for a Viterbi decoder implementation however is the code constraint length k, as the decoder complexity increases exponentially with k. This limits most Viterbi decoder implementations to codes of relatively small constraint lengths.

2.4 Convolutional codes performance with Viterbi algorithm

To implement a convolutional coding system, the appropriate code parameters, R and k, should be decided first. In addition, the design decisions for the decoding system, such as the quantization levels etc., are also crucial. This requires a knowledge of the code performance with the particular parameters. The most useful techniques for estimating the performance of convolutional codes are union bounds and computer simulation as explained in the following sections.

2.4.1 Error event and union bounds

When decoding a convolutional code with Viterbi algorithm, an error event occurs at time i if the Viterbi decoder chooses an incorrect path which has a smaller Hamming distance over the correct path. This is shown in Figure 2.13.

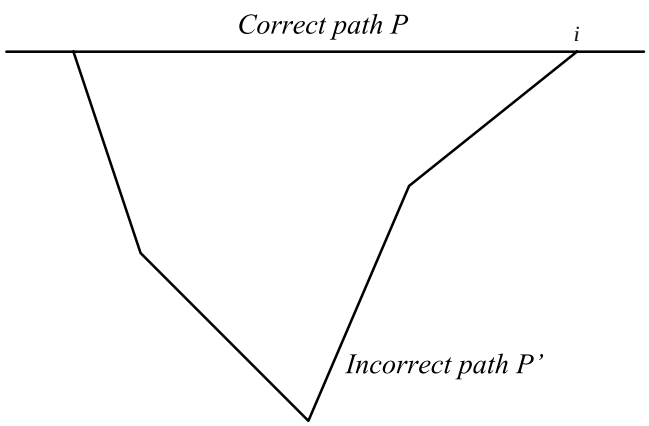


Fig. 2.13: The error event with Viterbi decoding.

The probability of an error event, P_e, at time i is bounded by the sum of the error probabilities for all possible paths beginning and ending in the correct path.

$$P_e < \sum_{d=d_{free}}^{\infty} n_d P_d \tag{2.8}$$

where n_d is the number of possible paths of weight d merging with the all-zero path, and P_d is their probability. The d_{free} is the minimum Hamming distance between two paths which begin and end in the same state.

The union bound on the bit error rate, P_b, is obtained by summing the input sequence weights w_d corresponding to these paths, which is given by

$$P_b < \frac{1}{m} \sum_{d=d_{free}}^{\infty} w_d P_d \tag{2.9}$$

for a $R = m/n$ code [7]. In practice, a finite number of w_d are computed to give an approximated union bound value. The accuracy of this approximation depends on the number of w_d samples used. The more the samples the closer the approximation is to the real value. Figure 2.14 compares the BER results from union bound calculation and simulation for $R = 1/2$, $k = 7$ code with hard-decision Viterbi decoding. The line marked with crosses shows the BER from the union bound approximation with seven w_d samples. The line marked with stars shows the BER from Monte Carlo simulations with a maximum 100 errors. The simulation results were obtained from using Matlab functions and the program is listed in Appendix A.1. This figure indicates that at high Eb/No levels, the simulated BER is smaller than the union bound by as much as half at $Eb/No = 3.5$ or $4.5dB$. However, as the noise level goes down, e.g. for Eb/No higher than $5dB$, the union bound becomes close to the simulation result with an accuracy within a small fraction of a decibel. The results suggest the approximation of union bound can be used in estimating convolutional code BER performance, especially for performance comparison where the BER difference other than the absolute BER value

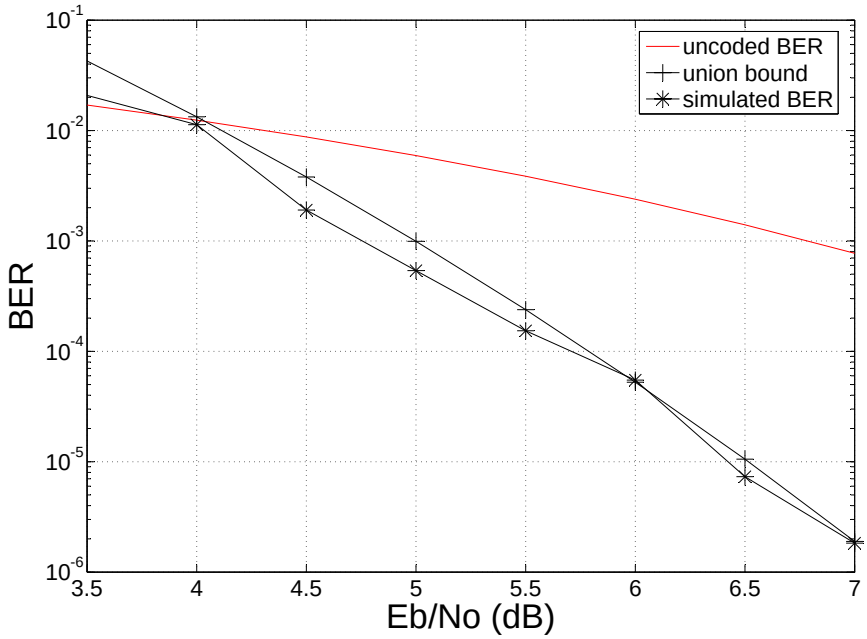


Fig. 2.14: Union bound and simulated BER for R=1/2, k=7 code with hard-decision Viterbi decoding and QPSK (Quadrature Phase Shift Keying) modulation.

is the major concern. Therefore, the approximated union bounds are used to show the possible BER performance of the convolutional codes with different parameters in the following analysis.

2.4.2 Convolutional code performance

As discussed before, the performance of convolutional codes are determined by the code rate R, the constraint length k, hard or soft decision, the quantization levels, the generator polynomials and the decoding method. In practice, only the first three parameters are of normal concern. To analyse the effects of these parameters, the union bounds of the Viterbi algorithm with different parameter sets are compared.

Code rate R. The code rate of a convolutional code determines the entropy of the encoded sequence and the distance properties, and thus affects the BER performance. Figure 2.15 is obtained from the simulations using the Matlab program shown in Appendix A.2 and compares the union bound with soft-decision of the code at rate 1/3, 1/2, and 2/3. These codes are produced by the encoders with the same amount

of memory to give the same number of encoder states. This provides approximately the same decoding complexity for each information symbol. As expected, the BER

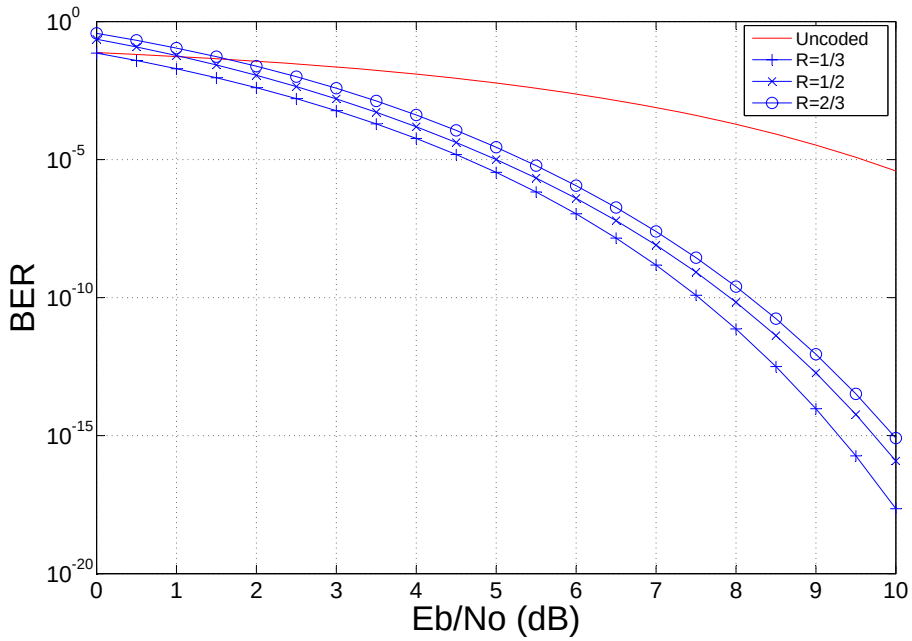


Fig. 2.15: Union bounds for R=1/3, 1/2, and 2/3, code with soft-decision Viterbi decoding and QPSK modulation.

performance illustrates a drop as the rate increases. This indicates a tradeoff relation between the information transmission rate and the BER performance. One fact shown by this figure is that the differences of the BER figure of two codes are evenly distributed over Eb/No levels. This suggests the code rate or the average entropy of a convolutional code has a constant influence on the error probability of the code; this is because of the linear relationship of the redundancy with the distance properties. At $Eb/No = 3.5dB$, the BERs are $1.9e^{-4}$ and $5.2e^{-4}$ for rate 1/3 and 1/2 codes, respectively. The redundancy of rate 1/3 code is about 33% more than the 1/2 code. As the BER of 1/3 code is 63% lower than 1/2, it gives an average BER improvement figure of 2% for every percent increase in redundancy. The same figure can also be obtained by comparing the 1/2 with 2/3 codes. Therefore, it may be concluded that for a convolutional code if the redundancy doubles, the BER performance will be improved by around a factor of four.

Constraint length k. Increasing the constraint length at a fixed code rate also improves the BER performance. This is shown by Figure 2.16 which is from the Matlab simulation in Appendix A.3. The interesting point shown by Figure 2.16 is that

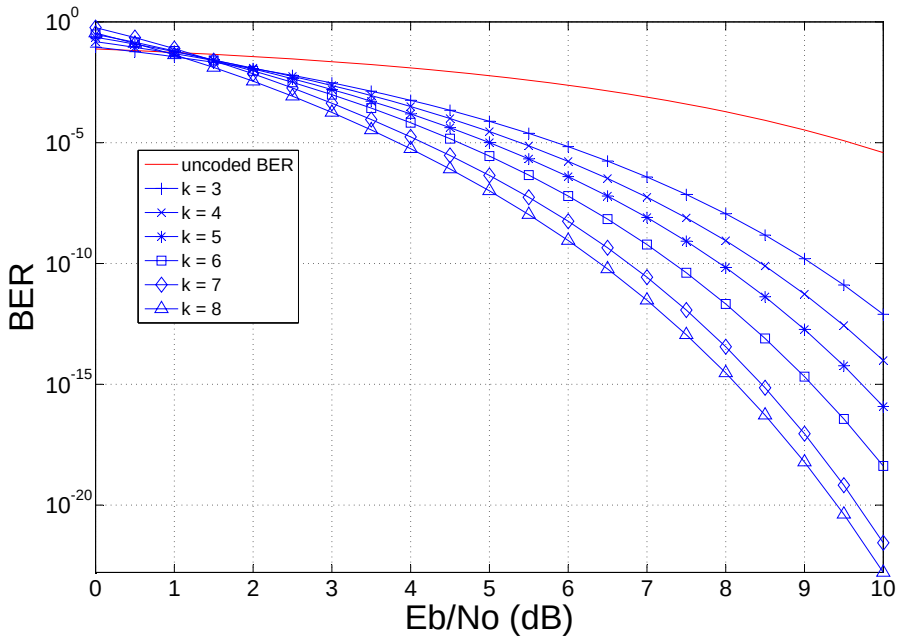


Fig. 2.16: Union bounds for R=1/2, k=3, 4, 5, 6, 7, and 8 code with soft-decision Viterbi decoding and QPSK modulation.

the improvement of BER performance with a longer constraint length is not evenly distributed over different Eb/No levels. This is very different comparing with the Figure 2.15. For example, at $Eb/No = 2.5dB$, the BER of the $k = 3$ code is $6.1e^{-3}$ whereas the BER is $8.4e^{-4}$ for the $k = 8$ code. This indicates a 7 times better BER performance of the $k = 8$ code. However, the performance of $k = 8$ code becomes about 4×10^{10} times better than the $k = 3$ code at a Eb/No level as low as $10dB$. This is a very significant performance improvement. Therefore, a longer constraint length is actually preferrable for a decoder design as long as the increased complexity is still affordable.

Moreover, this result also shows that in terms of improving BER performance, to increase the constraint length is much more efficient than simply adding more redundancy to the code word. The key to an ideal, error free communication is the perfect

constructing of the information inter-lock. The longer the constraint length the more the information are inter-locked and protected by each other. Thus, the optimum error protection capability can be achieved.

Hard or soft decision. Soft-decision provides significantly higher BER performance for the decoding system with the same code. This result is obtained by the simulation program in Appendix A.4 and shown in Figure 2.17. The BER of soft-decision, $R = 1/2$ code gives a 4×10^4 times lower BER at $Eb/No = 5.5dB$ than the hard-decision as is shown in Figure 2.17. Since implementing soft-decision has relatively

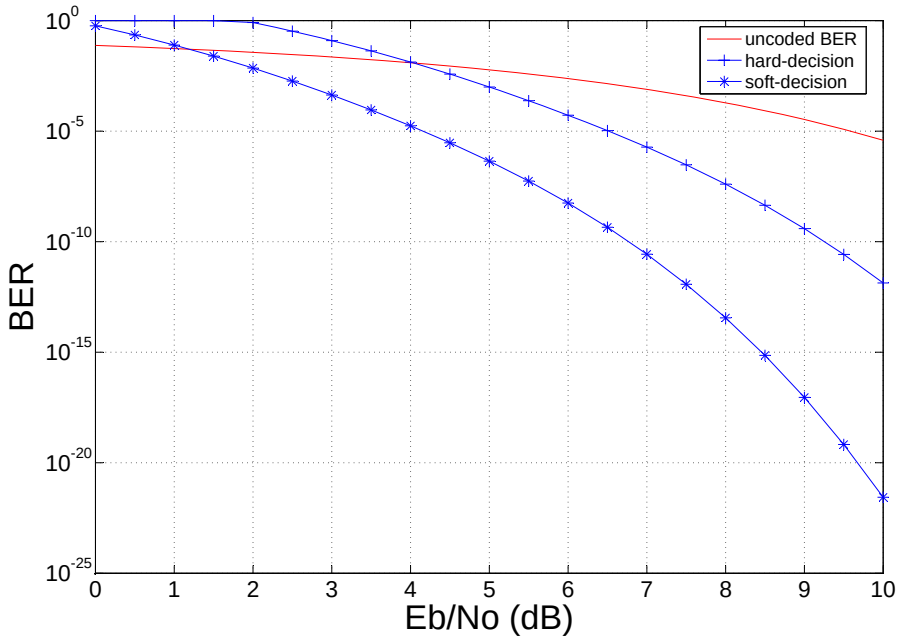


Fig. 2.17: Union bounds for R=1/2, k=7 code with both Soft and Hard decision Viterbi decoding and QPSK modulation.

small impact on the decoder complexity, most of the decoding systems incorporate the soft-decision capability.

2.5 summary

In this chapter, the convolutional code is discussed extensively from the basic concepts, the code structures, and the distance properties to the BER performance. In terms

of the distance properties, a good convolutional code has a d_{free} as large as possible but keeps the distance between the corresponding input sequence as small as possible. Influenced by the code rate, the constraint length and the generator polynomials, the distance properties determine the performance of the convolutional code. The concept of union bound provides a solution to estimate the code performance with the ML decoding.

The Viterbi algorithm is a simple iterative process implementing the concept of ML decoding. The code performance analysis in this chapter indicates a simple fact that to improve one feature of the convolutional code there must be some losses in another feature. Although increasing the constraint length gives superior BER performance, it is not necessarily the best solution for lowering BER. For a coding system design, the decoding complexity, the system power consumption, the delay, and the information transmission rate, etc., should all be considered. The optimum design is the design with a perfectly balanced position in terms of all these features as well as meeting the design requirement. In the next chapter, the Viterbi decoder design is discussed with a focus on its complexity and power consumption.

3. VITERBI DECODER AND ITS POWER DISSIPATION

Designing a Viterbi decoder involves many considerations, such as the decoding accuracy, the design complexity, the power consumption, the throughput, the output delay, etc.. To meet the requirements, the design has to be balanced in terms of these features. This requires a fundamental knowledge of the characteristics of the standard Viterbi decoder and how these characteristics change with the design.

3.1 Viterbi decoder design

The classical Viterbi decoder design is a straightforward implementation of the basic processes of the Viterbi algorithm. The design consists of three functional units, as shown in Figure 3.1.

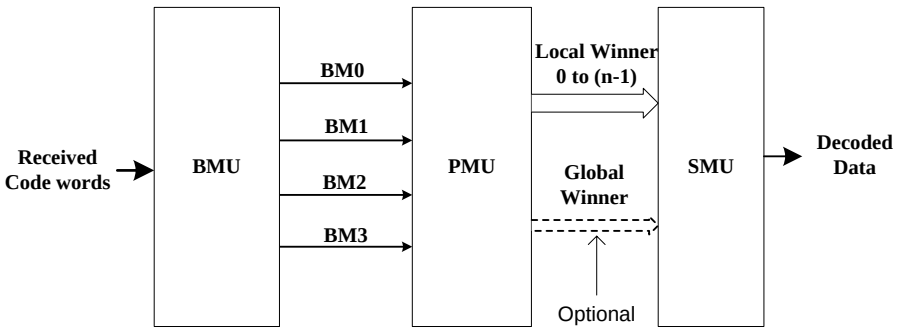


Fig. 3.1: Classical three functional block of a rate 1/2 Viterbi decoder design.

1. The BM Unit (BMU) which calculates the BMs;

2. The Path Metric Unit (PMU) includes a number of Add Compare Select Units (ACSU) which add the BMs to the corresponding PMs, compares the new PMs,

and select the PMs indicating the most likely path; at the same time, the PMU passes the associated survivor path decisions, called local winners, to the Survivor Memory Unit (SMU);

3. The SMU which stores the survivor path decisions; then the accumulated history in the SMU is searched to track down the most likely path so that the decoded sequence can be decided.

3.1.1 BMU design

The BMU is the simplest block in the Viterbi decoder design. However, the operation of BMU is crucial as it is the first stage of the Viterbi algorithm and the consequent decoding process depends on all the information it provides. In a hard-decision Viterbi decoder, the BMU design is straightforward since the BMs are the Hamming distances between the received code words and expected branches. For a soft-decision Viterbi decoder, the received code words are quantised into different levels according to the signal strength then the BMU maps the levels of code words into BMs according to their likelihood.

In hard-decision the Hamming weight of the code word is used as the branch metric, which is simply the number of positions in which the received code word differs from the ideal code word. The case of soft-decision can be derived from the generalised unquantised (analogue) channel.

For an unquantised channel, assume binary antipodal signalling is used with a convolutional code of rate m/n. If a code word S, which consists of n symbols, $x_0\ x_1 \cdots x_{n-1}$, is transmitted through the channel, the decoder receives R which is a sequence of n sampled actual voltages, $r_0\ r_1 \cdots r_{n-1}$, from the filter. The conditional probability of sending S and receiving R is [7]

$$P(S|R) = P(R|S)\frac{P(S)}{P(R)}. \tag{3.1}$$

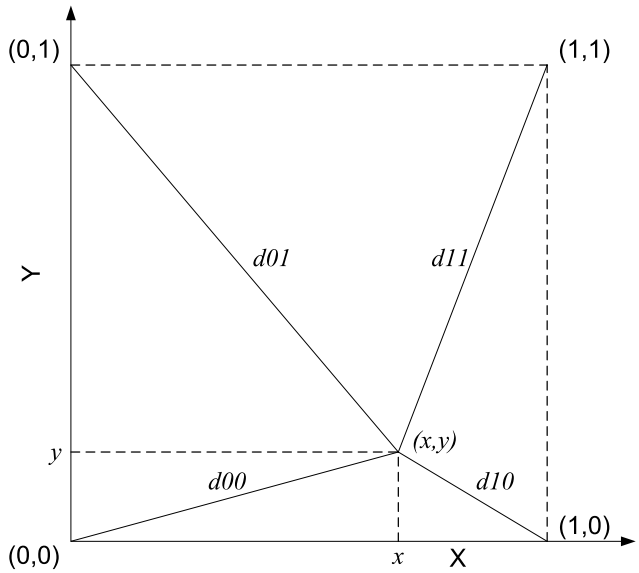
If the transmitted code words have an equal probability, an optimum decoder identifies the S which maximises $P(R|S)$ so that the maximum $P(S|R)$ can be achieved. Since a code word has n symbols, for the Gaussian noise with zero mean and variance $\sigma^2 = N_o/2$ where N_o is the noise power spectral density, $P(R|S)$ becomes the product of n Gaussian density functions of each symbol. As given in [7]

$$P(R|S) = \prod_{i=0}^{n-1} P(r_i|s_i) = \left[\frac{1}{(\pi \times N_o)^{1/2}}\right]^n \times exp\left[-\sum_{i=0}^{n-1} \frac{(r_i - s_i)^2}{N_o}\right] \tag{3.2}$$

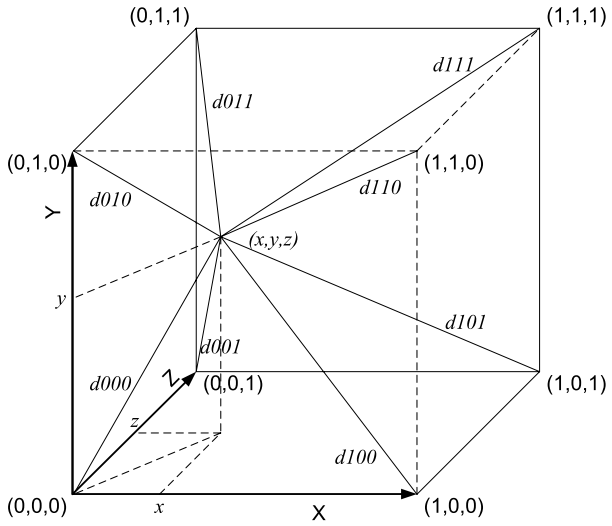
For a specific noise level, the $P(R|S)$ is maximised when

$$d^2 = \sum_{i=0}^{n-1} (r_i - s_i)^2 \tag{3.3}$$

is minimised, where d^2 is the squared Euclidian distance between the hypothesized sequence and the received signal. For an unquantised channel, d^2 can be used as the measurement of the unlikelihood of the code word branch, e.g. the branch metric, since a minimum value of d^2 indicates the most likely branch and its accumulated value indicates the most likely path. This squared Euclidian distance is defined in [7] as the generalised concept of the distance between the received and ideal code words. For the received signal from the additive white Gaussian noise (AWGN) channel, the signal level of each symbol is independent. Thus, a code word which consists of n symbols forms an n-dimensional space. For instance, Figure 3.2(a) shows the distances of the code word with 2 symbols, X and Y. There are four ideal code words, $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$, which are located at the four corners in this 2-dimensional space. The received signal (x, y) are unquantised and represent the two received code word symbols having the value range from 0 to 1. Due to the noise the received signals do not correspond to any of the ideal points. Thus, the distance labelled with $d00$, $d01$, $d10$, and $d11$ are the Euclidian distances; d, between (x, y) and the four ideal points, where, for example, $d00^2 = x^2 + y^2$. In the 3-dimensional space formed by



(a) The distance of a 2-symbol code word represented in a 2D space.



(b) The distance of a 3-symbol code word represented in a 3D space.

Fig. 3.2: The Euclidian distance between the hypothesised code word and the received signal.

3-symbol code words, as shown in Figure 3.2(b), there are 8 Euclidian distances, $d000$, $d001$, $d010$, $d011$, $d100$, $d101$, $d110$, and $d111$, for the received signals (x, y, z) and the distance $d000$ becomes $d000^2 = x^2 + y^2 + z^2$.

In digital communication systems, it is not possible to process the actual analogue voltages r_i; instead, the sampled voltages are quantised into m-bit numbers. In hard-decision, a signal is quantised into a one-bit binary number. In receiving the code word 00, for instance, the $d11^2 = (1-0)^2 + (1-0)^2 = 1 + 1 = 2$ and is consistent with the Hamming distance described above. Other than single bit, three-bit quantisation is the most commonly used scheme in communication system designs. Figure 3.3 illustrates the three-bit quantisation for the 2-symbol code words. As shown in Figure 3.3, the

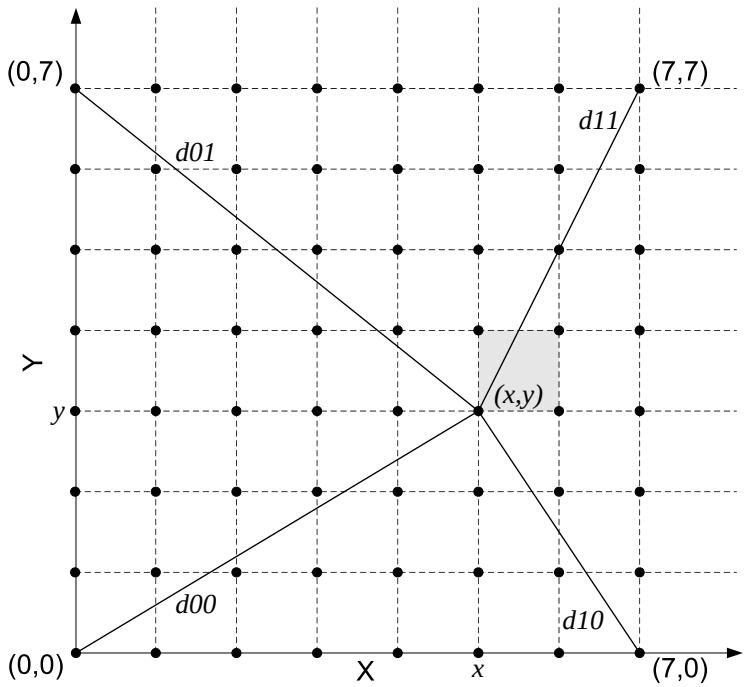


Fig. 3.3: Three-bit quantisation in a 2-dimensional space.

2-dimensional space is partitioned into $2^3 \times 2^3 = 64$ regions with the ideal code words 00, 01, 10, and 11, located at positions $(0, 0)$, $(0, 7)$, $(7, 0)$, and $(7, 7)$, respectively. The actual voltage within each region is approximated to the point, marked in black dots, with the smallest X and Y values. For example, the received signals within the

shaded region in Figure 3.3 are approximated to the point (x, y). Then the squared Euclidian distances for (x, y) can be used as the approximated distance of the received signal. Although a squared Euclidian distance is a simple addition of numbers, it still involves squares of the quantised signal values and causes engineering difficulties. Fortunately, the squared Euclidian distances of (x, y) can be linearly transformed into the Manhattan distance [7]

$$\begin{aligned} d00_{Mah} &= x + y \\ d01_{Mah} &= x + [(2^m - 1) - y] \\ d10_{Mah} &= [(2^m - 1) - x] + y \\ d11_{Mah} &= [(2^m - 1) - x] + [(2^m - 1) - y] \end{aligned} \tag{3.4}$$

by subtracting $(x^2 + y^2)$, dividing by $2^m - 1$ and then adding $x + y$, where m is the number of bits in the quantisation. Since the Viterbi algorithm is a linear process, using the Manhattan distance yields no accuracy degradation compare to the squared Euclidian distance, but simplifies the implementations. This can also be generalised to the squared Euclidian distance of any n-symbol code word, so that the Manhattan distance is used as the branch metric for a received code word.

Since the Manhattan distance is the addition of the distance in n independent directions, it can be further normalised in each direction. For the 2-symbol example shown in Figure 3.3, all the Manhattan distances of the point (x, y), listed in equation 3.4, can be simplified in the form $d_{Mah} = d_X + d_Y$, where the d_X and d_Y are distances on the X and Y axis, respectively. Assuming $x \leq (2^{m-1} - 1)$ d_X on the axis X can be subtracted by x, then the normalised distance on axis X between the symbol X and ideal symbol 0 is always zero; whereas the normalised distance to ideal symbol 1 is $(2^m - 1) - 2x$. Similarly distances on Y axis can be normalised to be either 0 or $(2^m - 1) - 2y$ when $y \leq (2^{m-1} - 1)$. Based on this, the branch weight scheme for the 2-symbol, 3-bit quantisation can be simplified as shown in Table 3.1. Therefore, a standard BMU design

Tab. 3.1: Branch weight scheme for 2-symbol, 3-bit quantisation.

Quantised level	**Weight referenced to 0**	**Weight referenced to 1**
0 (strongest 0)	0	7
1	0	5
2	0	3
3	0	1
4	1	0
5	3	0
6	5	0
7 (strongest 1)	7	0

assigns the weights to each symbol based on its quantised level and the weight scheme and adds the weights of each symbol together to make the branch metric. Because of its simple operation, the BMU in a Viterbi decoder is normally the simplest block and consumes much less power than the PMU and SMU blocks.

3.1.2 PMU design

The major task of the PMU is to calculate the metrics of the selected paths. These calculations are based on a generalised 2^m-to-2^m state transition diagram for a $R = m/n$ convolutional code encoder. Figure 3.4 shows the states transitions for a rate $R = 1/2$, $k = j$ code and is well known as the butterfly diagram. According to Figure 3.4, the

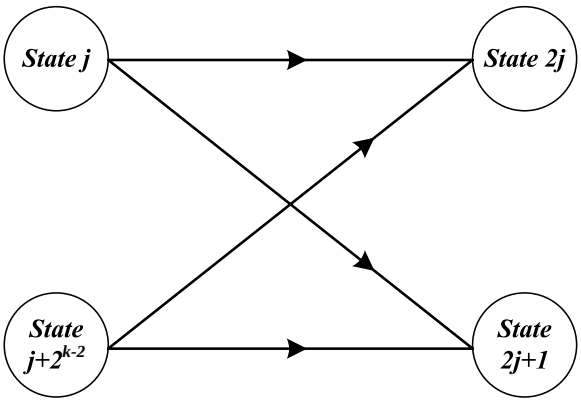


Fig. 3.4: The butterfly state transition diagram represents state transitions of a convolutional encoder of constraint lenght k.

path metrics for states j and $j + 2^{k-2}$ are added with two branch metrics to give two pairs of possible path metrics candidates for states $2j$ and $2j + 1$; the smaller ones of

each path metrics pair are then kept as the survivor path metrics and used for the next iteration of the calculations.

In a standard PMU, these add-compare-select operations are modular and performed by ACS units shown in Figure 3.5. Figure 3.5 shows a pair of path metrics S_i

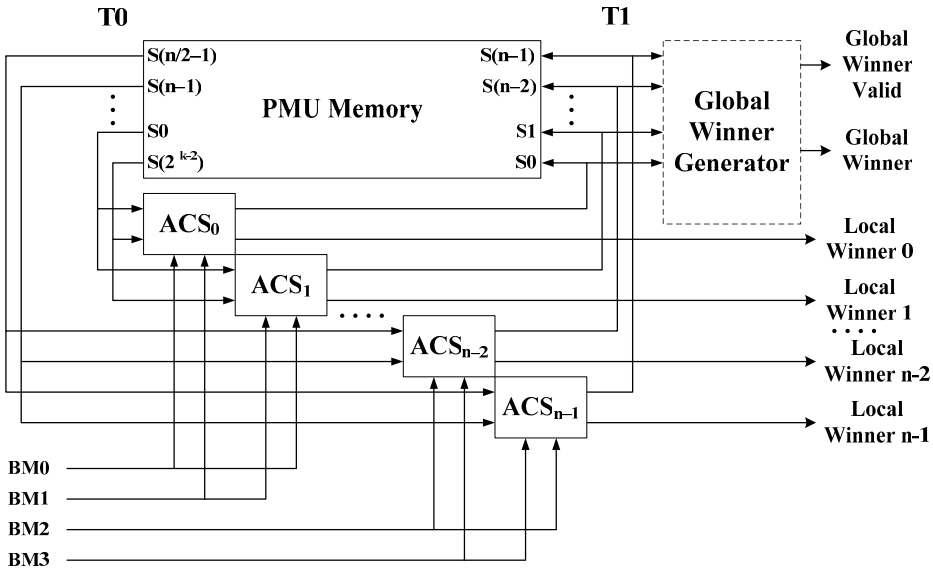


Fig. 3.5: A rate 1/2, n states *PMU* architecture shows the recursively calculations of state metrics value. The Global Winner Generator, shown in dashed lines, provides global winners information for a trace back *PMU*.

and $S_{(i+2^{k-2})}$ at time T_0 input to two ACS units representing the two candidates in the butterfly interconnection of state transitions described in Figure 3.4. These are added to the appropriate branch metric and compared. The selected new state metric is then output from each ACS unit and is written back to the PMU memory to become the current state metric in the next time slot, T_1. As well as the new state metric, each ACS also outputs a selection bit which indicates whether the selected branch was in the upper or lower position. These are shown as the local winner signals in Figure 3.5. The local winners are the most important information used for generating the output in the SMU.

Optionally, a global winner generator, marked in dashed lines in Figure 3.5, can be used in the PMU to identify the global winner which is the start point for a trace back in the SMU. The global winner generator shown in Figure 3.5 normally consists

of a comparator-tree structure which compares all state metrics and then outputs an index of the state with the lowest state metric value. The conventional way of finding the global winner is the major time overhead for the PMU design. However, from a power saving point of view, starting the trace back from the global winner state is more power efficient than starting from a random state, as far less time slots are required to be stored in the memory in order to achieve the path convergence, as discussed in section 2.3.1. Moreover, since starting trace backs at global winner states are likely to just extend existing trace backs, less transitions can be expected and thus, this reduces trace back power dissipation. For these reasons, the PMU design is important for the power efficiency of the whole Viterbi decoder since it not only determines just the power consumption of the PMU but also affects the power efficiency of the SMU.

3.1.3 SMU design

In the decoder, the SMU is the block which recovers the received data based on all the information from the PMU. It also consumes a large amount of power. For a trace back SMU with RAMs, up to 63% overall power is consumed [27] as it requires a large memory to store the local and global winners information as well as complex logic to generate the decoded data.

Two major types of SMU implementation exist: Register Exchange [28], [29], [30] and Trace Back [29], [31], [32].

Register exchange approach

Figure 3.6 [28] illustrates the principle of a 4 state register exchange architecture. In this architecture, a register is assigned to each state and contains decoded data for the survivor path from the initial time slot to the current time slot. As illustrated in Figure 3.6, the ideal path is indicated with bold arrows. According to the local winner of each state, the register content is shifted into another state register and appended with the corresponding decoded data. For instance, at time slot T_1 the survivor branch

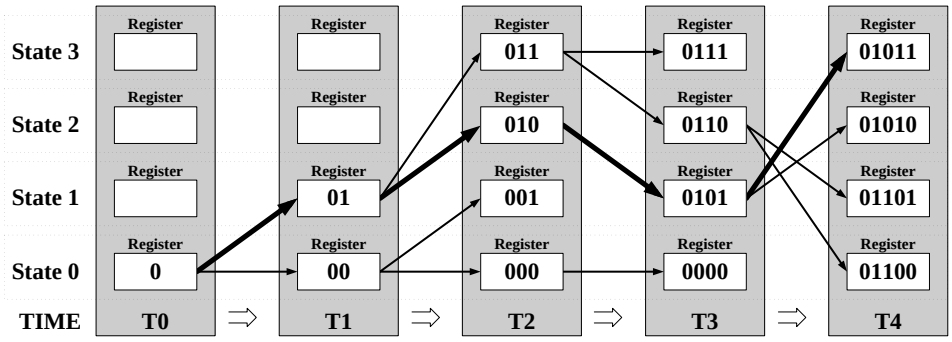


Fig. 3.6: A 4-state register exchange implementation of the *SMU* design. The bold arrows indicate the ideal path of the encoder states.

for state 1 is from state 0 at T_0; therefore, the initial content of the state 0 register, which is a '0', is shifted into state 1 register at T_1 and the corresponding decoded data for the survivor branch, which is a '1', is appended to it.

Registers on the ideal path, as shown in Figure 3.6, spread their contents to other state registers as time progresses due to the nature of ACS process. Thus, at the end of time slot T_4, the state registers all contain the bit(s) from the same source registers, which is the state 1 register at time T1. As shown in Figure 3.6, the two most significant bits of each register at time slot T_4 is "01". Therefore, this is the decoded output for timeslots T_0 and T_1.

The register exchange approach is claimed to provide high throughput [28], [30], as it eliminates the need to trace back since the state register contains the decoded output sequence. However, it is obviously not power efficient as a large amount of power is wasted by moving data from one register to another. In addition as D-type flip-flops rather than transparent latches need to be used to implement the shift registers although the amount of data which needs to be held to determine the output is identical to that required for trace back approach. This all leads to relatively high power consumption.

Trace back approach

The trace back approach is generally a lower power alternative to the register exchange method. In trace back, one bit for the local winner is assigned to each state to indicate if the survivor branch is from the upper or the lower position. Using this local winner, it is possible to track down the survivor path starting from a final state and this search is enhanced by starting from a global winner state as previously discussed.

Figure 3.7 shows a trace back SMU architecture adapted from the architecture described in [1] which used global winner information. Here, local winners are stored

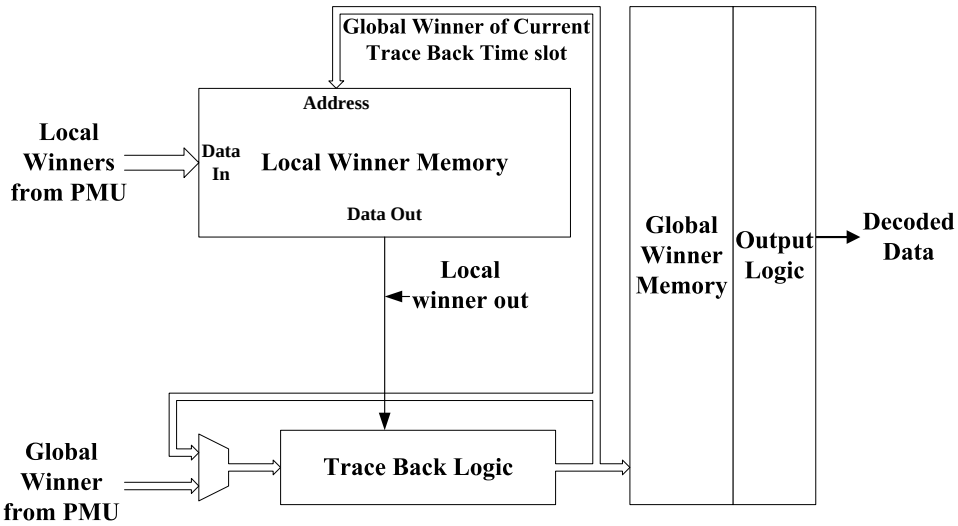


Fig. 3.7: A possible trace back *SMU* implementation using memory. It also requires global winner information in order to reduce the trace back depth.

in the local winner memory. Trace back is started at the global winner from the PMU, which is used as an address to read out the local winner of the global winner state. Then, in the trace back logic the previous global winner in the trace back is produced by shifting the current global winner one place to the right and inserting the read out local winner into the most significant bit position; this arithmetic relationship between parent and child states derives from the butterfly connection shown in Figure 3.4. This new global winner can then be stored into the global winner memory to update the global winner existing at that time slot. The process repeats with the updated global winner reading out its local winner which is used to form the global winner for

the previous time slot. This process continues until the global winner formed agrees with that stored or it reaches the oldest time slot [1]. In the output logic, shown in Figure 3.7, the decoded output can be obtained from the least significant bit of the global winners stored in the global winner memory.

As described in the last section, local and global winners are stored in memory. So for each trace back, local winners are repeatedly read out from the local winner memory and new global winners are written back to the global winner memory. This results in complex read/write control mechanisms. Furthermore, unless flip flop storage is used then multi-port SRAM blocks are required as seen in previous implementations [33], [34]. Moreover, it is preferable to run trace backs in parallel as an incorrect trace back may damage a "good" path and it needs a new trace back to correct this as soon as possible. It has been suggested in [35] that the read-write-based trace back also has a serious speed overhead due to the need to access multiple memory pointers. Therefore, reducing the complexity of the trace back logic and memory, increasing the trace back throughput, and reducing the SMU power consumption are all current research issues in Viterbi decoder designs [28], [33] and [34].

Many approaches have been proposed attempting to address these issues, e.g. increasing the number of pointers for parallel trace backs, decreasing the memory access time of the read operation, or increasing the access rate of the read operation in a time-multiplexed method [29], [31], [36]. However, none of them change the fundamental read-write architecture in the trace back implementations, so have only limited success in solving these problems.

3.2 Power dissipation in the Viterbi decoder

Over the past 15 years, Complementary Metal Oxide Silicon (CMOS) technology has played an increasingly important role and already occupies the major place in the global integrated circuit (IC) industry [37]. Most new VLSI designs are implemented

in CMOS technology because of its high performance, high packing density, low power attributes, and relatively low cost.

3.2.1 CMOS Circuitry power dissipation

In a CMOS circuit, power dissipation is comprised of three major components [37] [38]:

$$\begin{aligned} P_{avg} &= P_{switching} + P_{short} + P_{leakage} \\ &= fC_L V_{dd}^2 + I_{SC} V_{dd} + I_{LK} V_{dd} \end{aligned} \tag{3.5}$$

The first two terms are the switching and short circuit power dissipation, $P_{switching}$ and P_{short}, that are known as the dynamic power dissipation caused by switching activity, In equation 3.5 f is the switching frequency, C_L is the load capacitance, I_{SC} is the short circuit current and V_{dd} is the supply voltage. The third term of the equation is the leakage power consumption, known as the static power dissipation and is caused by leakage current, I_{LK}.

Switching power dissipation

A normal CMOS gate consists of two parts: a pull-up network made of PMOS transistors connected between the positive supply voltage and the output node, and a pull-down network made of NMOS transistors connected between the output node and ground [39]. Because of the advantage of PMOS in presenting logic 'one' and the advantage of NMOS in presenting logic 'zero', they together can build complementary logic gates. In an inverter circuit, for example, as the output changes from logic 'zero' to 'one', current flows from power supply to various capacitances, which are C_L in total, and charge them to V_{dd}. This charge consumes an energy of $fC_L V_{dd}^2$, half of which is stored in the output capacitor and half is dissipated in the resistance of the PMOS transistor [38] [39]. When the output changes from logic 'one' to 'zero', the stored energy will be dissipated in the resistances of the NMOS transistors, although there is no

energy drawn from the supply. If the frequency of the power consuming transitions (0 to 1) is f, the power drawn from the supply is $fC_LV_{dd}^2$. The switching power dissipation is the major part of the CMOS power consumption, so much research has been carried out and numerous methods are suggested for minimising it. From a designer point of view, therefore, reducing the switching power dissipation by minimising the switching activity in the design is the major method of achieving power efficiency. Methods of reducing switching power dissipation in the Viterbi decoder design are the emphasis of this thesis.

Short-circuit power dissipation

The analysis in the previous section is based on the assumption that only one of the transistors in the inverter circuit is conductive at any time. However, in practice there is a short period of time during each transition (either 0 to 1, or 1 to 0) when both PMOS and NMOS transistors are conducting; this is caused by the finite rise and fall times of the input waveforms. This period is determined by the voltage of the input. For the inverter circuit, when the condition $V_{tn} < V_{in} < V_{dd} - |V_{tp}|$ holds for the input voltages, where V_{tn} and V_{tp} are the NMOS and PMOS threshold voltages, the NMOS and PMOS devices are simultaneously on and generate a conductive path between V_{dd} and Gnd [40]. The short-circuit current, I_{SC}, therefore, draws an energy of $I_{SC}V_{dd}$ from the supply.

Because the short-circuit currents are significant when the rise/fall time at the input is much larger than the output rise/fall time, it is very important to minimise the transition times of input signals, in order to minimise the short-circuit period. In the case that input and output have equal edge times, the short-circuit power consumption is normally less than 10% of the total dynamic power dissipation [37]. Furthermore, as the short-circuit current occurs during each transition in a circuit, it is also important to minimise the numbers of transitions of the input. An interesting point here is that if the supply Vdd is lower than the sum of the thresholds of the transistors, $V_{dd} < V_{tn} + |V_{tp}|$,

the short-circuit current will never occur because both NMOS and PMOS devices will not be conductive at the same time for any value of input voltage [40].

Leakage power dissipation

Ideally, during the time when the state of the output, either zero or one, is unchanged, there is no current in the circuit and thus no energy will be dissipated. In practice, however, there are always small currents within the CMOS circuit even when devices are off. Two types of leakage currents occur: reverse-bias diode leakage current, which occurs due to parasitic diodes which form between areas of diffusion and the substrate [37], and subthreshold leakage current which occurs due to carrier diffusion between the source and the drain [40]. Although these power consumptions are relatively small when circuits are fully active, in systems, such as mobile phones, where large amount of time are spent in stand-by mode, the leakage power consumption can be a major problem.

3.2.2 Design flow and power estimation

The switching power dissipation, $fC_LV_{dd}^2$, also applies to the dynamic power consumption in a FPGA. Since the dominant part of CMOS power dissipation is the switching power dissipation, the estimated dynamic power consumption of a FPGA designs indicates the relative switching power dissipation of the CMOS design and thus, can be used for estimating the power consumption of Viterbi decoder designs with CMOS technology.

In this research, designs are completed with the Integrated Software Environment (ISE), which is a software suite developed by Xilinx that allows designers to take their designs from design entry through FPGA device programming. The ISE manages and processes a design through the following steps in the ISE design flow.

Design Entry

Design entry is the first step in the ISE design flow. During design entry, the design

source files can be created based on the design objectives using a Hardware Description Language (HDL), such as VHDL, Verilog, or ABEL, or using a schematic. Multiple formats for the lower-level source files are also supported in design entry.

Synthesis

After design entry and optional simulation, Xilinx Synthesis Technology (XST), integrated in ISE, synthesizes VHDL, Verilog, or mixed language designs to create Xilinx-specific netlist files. Then they are accepted as input to the implementation step.

Implementation

After synthesis, ISE design implementation converts the logical design into a physical file format that can be downloaded to the selected target device. The implementation process includes four major steps: ***Translate***, which merges the incoming netlists and constraints into a Xilinx design file; ***Map***, which fits the design into the available resources on the target device; ***Place and Route***, which places and routes the design to the timing constraints; ***Programming file generation***, which creates a bitstream file that can be downloaded to the device

Verification

A design can be verified at several points in the design flow. The integrated ISE simulator or ModelSim software can be used to verify the functionality and timing of a design or a portion of the design. These simulators interpret VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. In-circuit verification can also be carried out with the Chipscope software, also provided by Xilinx, after programming the FPGA device.

Device Configuration

After generating a programming file, it is downloaded from a host computer to a Xilinx device on a development board.

The XC4VSX35 FPGA device is used for in-circuit verification and BER testing. This device belongs to the latest Virtex-4 FPGA family which is based on $90nm$ CMOS

technology and featured with various techniques. The major features of the Virtex-4 SX class devices are listed in Table 3.2 [41]. The designing and testing flow are shown

Tab. 3.2: Features of the Virtex-4 SX class devices.

Features \ Devices	**XC 4VSX25**	**XC 4VSX35**	**XC 4VSX55**
Logic Cells	23,040	34,560	55,296
Block RAM/FIFO w/ECC (18 kbits each)	128	192	320
Total Block RAM (kbits)	2,304	3,456	5,760
Digital Clock Managers (DCM)	4	8	8
Phase-matched Clock Dividers (PMCD)	0	4	4
Max Differential I/O Pairs	160	224	320
XtremeDSP Slices	128	192	512
Configuration Memory Bits	9,651,072	14,476,608	24,088,320

in Figure 3.8.

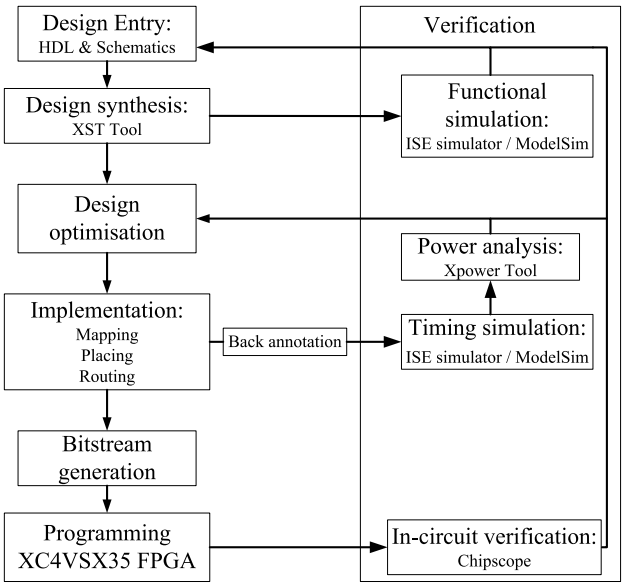


Fig. 3.8: Design and verification process of the FPGA implementations

3.2.3 Testing framework and noise generator design

The author has developed a test framework for the FPGA in-circuit simulations of a Viterbi decoder design. The top level of this test framework consisted of seven major blocks: a convolutional encoder, a uniform distributed random number generator, an

AWGN generator, an 8-level quantiser, a Viterbi decoder block and two error counters; this is shown in Figure 3.9. In the test framwork design, a random binary information

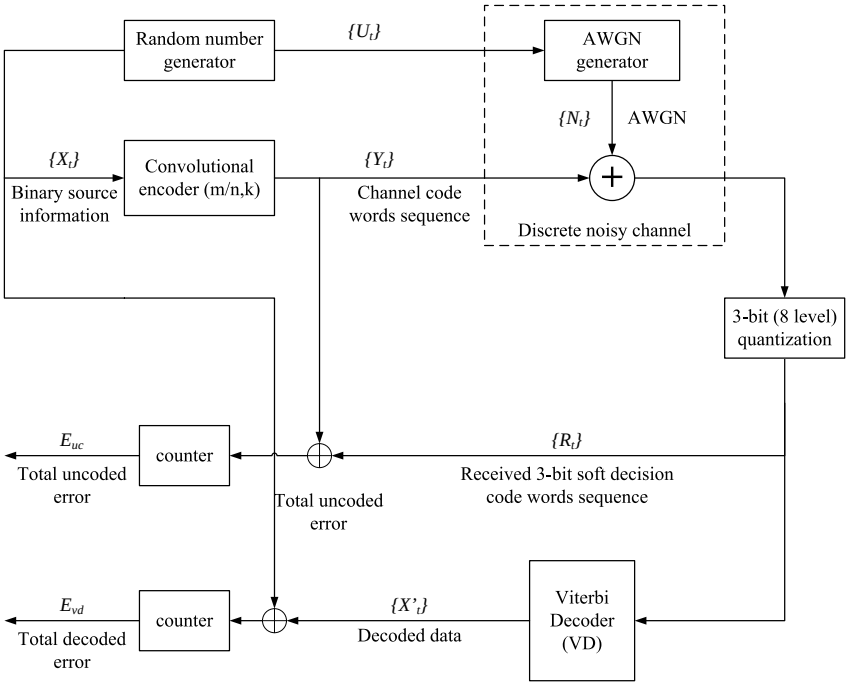


Fig. 3.9: The test framework of the FPGA implementaion.

sequence $\{X_t\}$ is encoded into a code word sequence $\{Y_t\}$ and sent to a discrete noisy channel at time t. In the channel, noise $\{N_t\}$ is added to $\{Y_t\}$ which gives the received signals. Then, an 8-level quantiser is used to produce the soft-decision $\{R_t\}$. Based on the soft-decision code words sequence $\{R_t\}$, the Viterbi decoder produces the estimated information sequence $\{X'_t\}$. By comparing $\{X_t\}$ and $\{X'_t\}$, the number of the decoded errors, E_{vd}, is obtained. To verify the noise generator, $\{Y_t\}$ and $\{R_t\}$ are compared to give the number of the uncoded errors E_{uc}. The AWGN generator design is described in Appendix B.

3.2.4 Viterbi decoder power consumption

The power consumption of Viterbi decoder designs on the FPGA were estimated by Xpower which offers detailed power analysis and estimation for programmable logic. XPower is integrated in ISE and allows designers to analyze total device power, power

per-net, routed, partially routed or unrouted designs. In a power test, the timing simulation of the post place and route design is carried out first to provide the circuit level transitions information, which is saved as a VCD file. Then, the design layout file and VCD file are imported into Xpower to give the estimated power consumption.

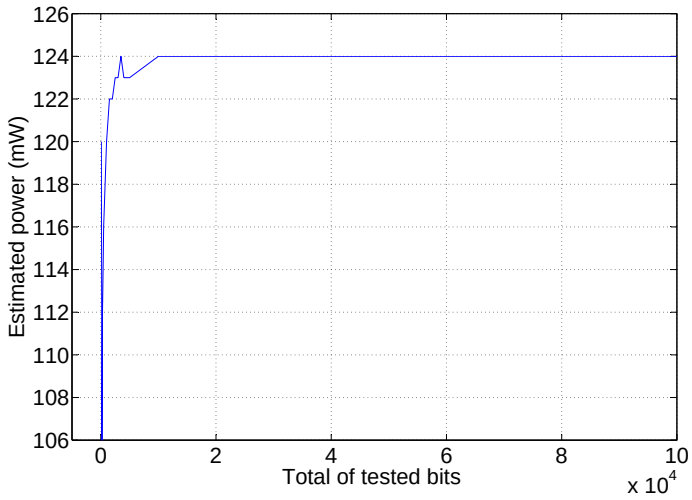
Power simulation setup

The most important parameter of the power simulation is the number of test samples. If the number of the decoded bits is too small, the estimated power of the decoder will not be accurate enough to indicate the decoder power dissipation with a much longer data sequence; if the number is too large, it will take an extremely long time for Xpower to calculate the power figure. Therefore, a set of simulations was first carried out with different numbers of test samples at a high noise level of Eb/No=0dB. Two standard decoders of $k = 3$ and $k = 7$ were used in the simulations. The results are shown in Figure 3.10(a) for the $k = 3$ decoder and Figure 3.10(b) for the $k = 7$ decoder. Figure 3.10(a) shows for the $k = 3$ decoder, the estimated total (Dynamic and Quiescent) power increases from $106mW$ to $124mW$ when the number of samples $n < 10,000$; however, for $n \geq 10,000$, the estimated power keeps constant. It is similar for the $k = 7$ decoder, as shown in Figure 3.10(b). The estimated power rapidly grows to $543mW$ at $n = 10,000$ and then it tends to reduce slowly. The tests with both decoders indicate the optimum samples number for the power simulation is $10,000$ since this is the minimum number of samples with which the estimated power is reasonably accurate. Thus, $10,000$ samples were used for all FPGA power simulations.

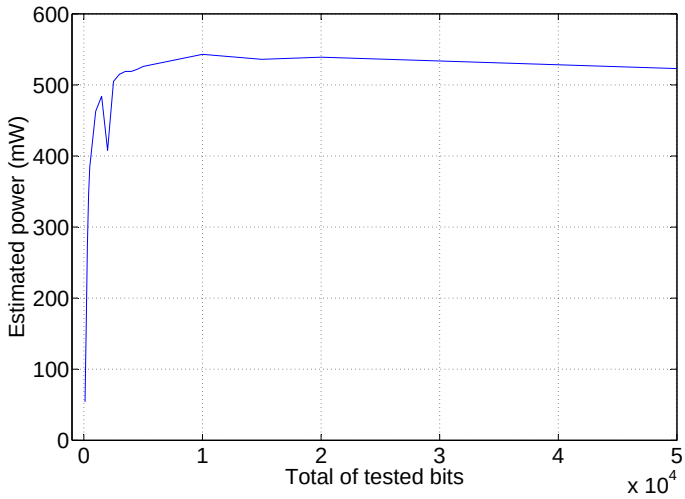
Viterbi decoder power dissipation

Eb/No test

Two standard Viterbi decoders of constraint lengths 3 and 7 are used in this test. Trace backs in these two decoders are started from the optimum states and have a length of 36 time slots. The dynamic power consumption of these decoders are measured at different



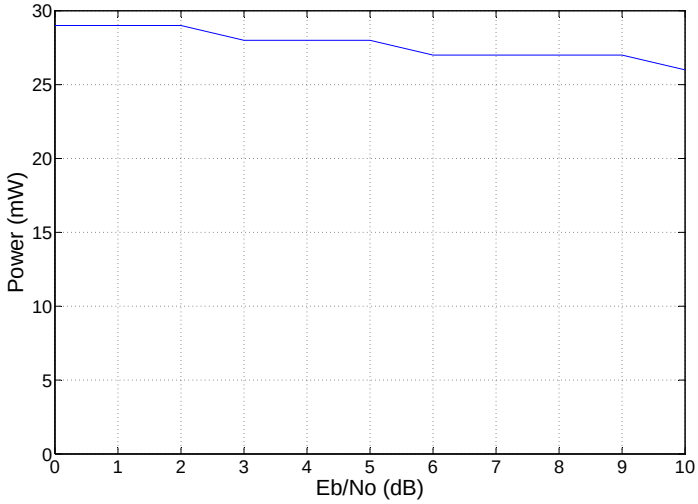
(a) The estimated total power consumption of the k=3 Viterbi decoder at 50MHz with different number of test bits at Eb/No=0dB.



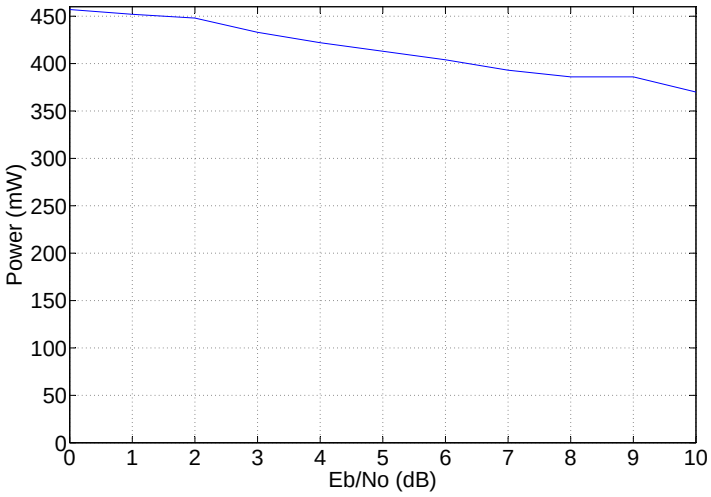
(b) The estimated total power consumption of the k=7 Viterbi decoder at 50MHz with different number of test bits at Eb/No=0dB.

Fig. 3.10: The optimum number of test bits for power simulations.

Eb/No conditions, from 0dB to 10dB, and the results are shown in Figure 3.11(a) and Figure 3.11(b). Both Figure 3.11(a) and Figure 3.11(b) indicate the dynamic power



(a) The estimated dynamic power consumption of the k=3 Viterbi decoder at different noise levels.



(b) The estimated dynamic power consumption of the k=7 Viterbi decoder at different noise levels.

Fig. 3.11: The Viterbi decoder power consumption at different Eb/No levels.

consumption of a standard Viterbi decoder decreases gradually with the noise strength. For the decoder of constraint length 3, the dynamic power consumption reduces from $29mW$ to $26mW$ which gives a 10% reduction at 10dB. With a constraint length 7, the dynamic power consumption of the decoder reduces from $457mW$ to $370mW$ and

the reduction is 19% at 10dB. The reductions of the dynamic power consumption are due to the reduced switching activities in the decoder (mainly in the SMU) and do not suggest any changes in decoder power efficiency. To analyse the energy efficiency of the Viterbi decoder, the power figures are averaged by the numbers of corrected errors in each second; this gives the average energy that the decoder consumes in correcting an error. Figure 3.12 illustrate the results of this efficiency analysis. It indicates the

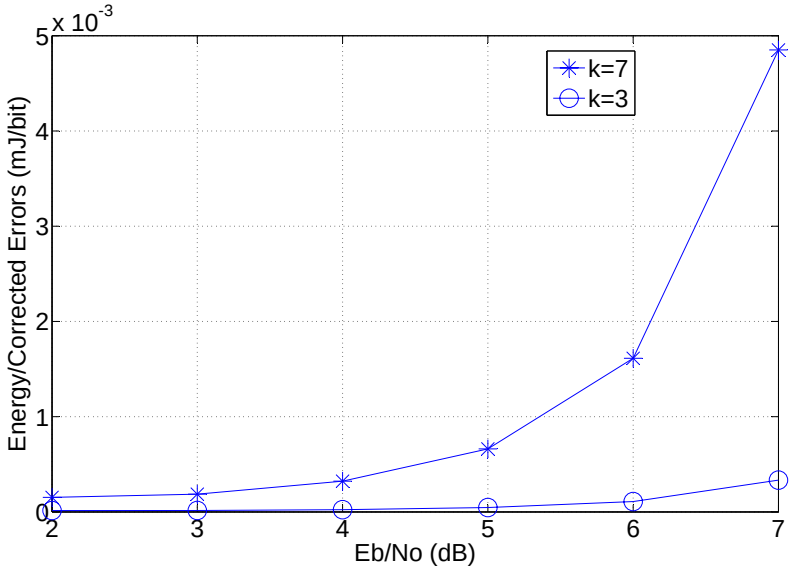


Fig. 3.12: Energy per corrected error of the Viterbi decoders of k=3 and 7.

exponential increase of energy consumption to correct a bit for both the Viterbi decoder k=3 and 7 as the noise level decreases. This, therefore, suggests the energy efficiency of a standard Viterbi decoder decreases significantly with the increase of Eb/No level. This is due to the reduction of the number of errors. When Eb/No increases, the number of errors falls dramatically. Although the power consumption reduces with the increase of Eb/No, it is not as fast as the decrease of the errors number; therefore, as Eb/No increases, more and more energy is wasted by the Viterbi decoder in processing an error free sequence and this reduces the energy efficiency of the decoder. Moreover, as Figure 3.12 indicates, for a longer constraint length decoder, the reduction of energy efficiency is more significant and thus, it is less energy efficient.

For the Viterbi algorithm, one issue which is commonly recognised by researchers is the exponentially increasing complexity from increasing the constraint length. The above analysis of the decoder energy efficiency reveals another major efficiency issue of the Viterbi algorithm: the energy efficiency of the Viterbi decoder dramatically reduces with the increase of Eb/No. This problem is actually caused by the noise independent nature of the Viterbi algorithm and is rarely recognised by the research community.

Constraint length relationship to power

Viterbi decoders with constraint lengths 3, 4, 5, 6, 7 and 8 are used in the test. The power consumptions are measured at Eb/No=2dB and the results are shown in Figure 3.13. Due to the exponentially increase of the complexity, as shown in Figure 3.13,

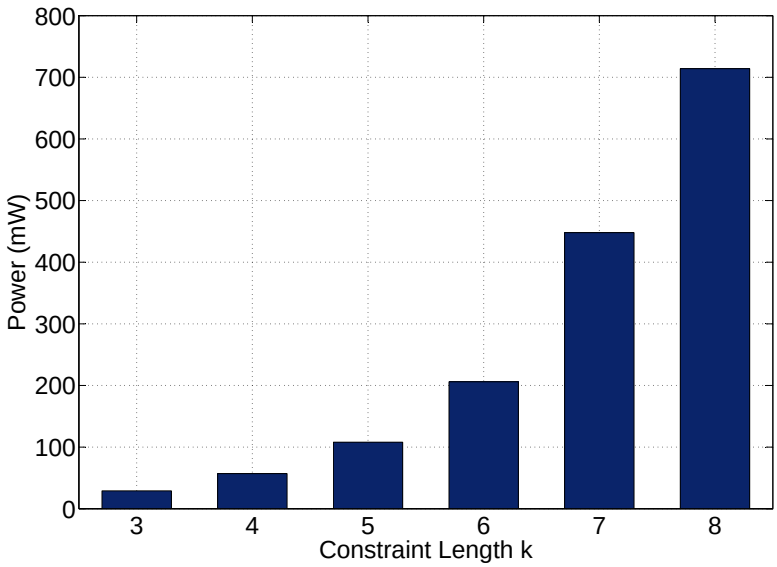


Fig. 3.13: The estimated dynamic power consumption of the Viterbi decoder with different constraint lengths at Eb/No=2dB.

the dynamic power consumption of the standard Viterbi decoder also increases exponentially, i.e. with the constraint length 8, the power is increased by a factor of 23 compared with that for constraint length of 3. However, since the decoder with longer constraint length can correct more errors, these power figures are not able to indicate the decoder power efficiency. The energy per corrected bit, therefore, is used again to analyse the decoder energy efficiency with different constraint lengths at $Eb/No = 2dB$;

and the results are shown in Figure 3.14. Figure 3.14 indicates a decrease of the decoder

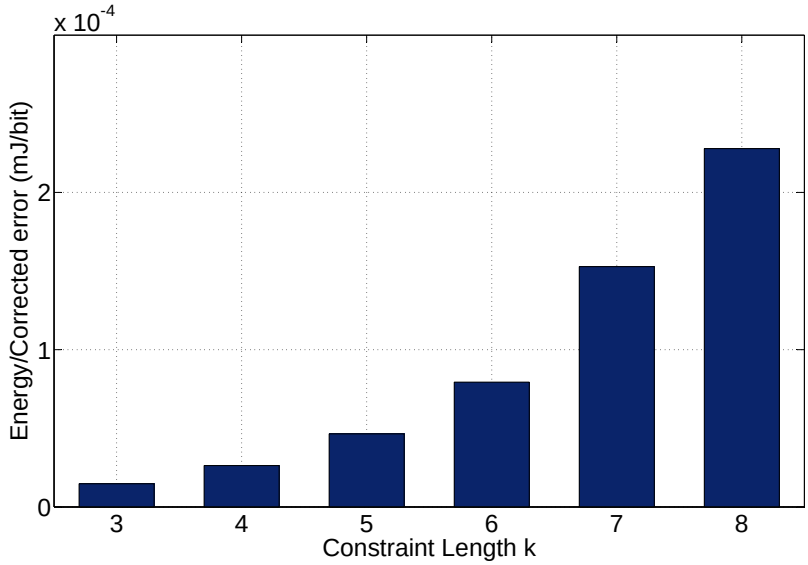
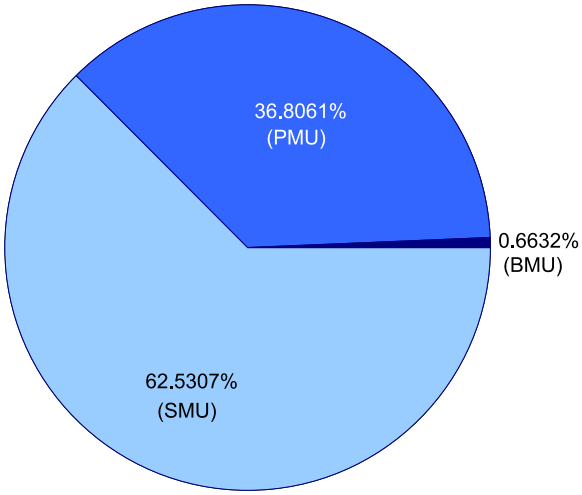


Fig. 3.14: Energy per corrected error of the Viterbi decoder with different constraint lengths at Eb/No=2dB.

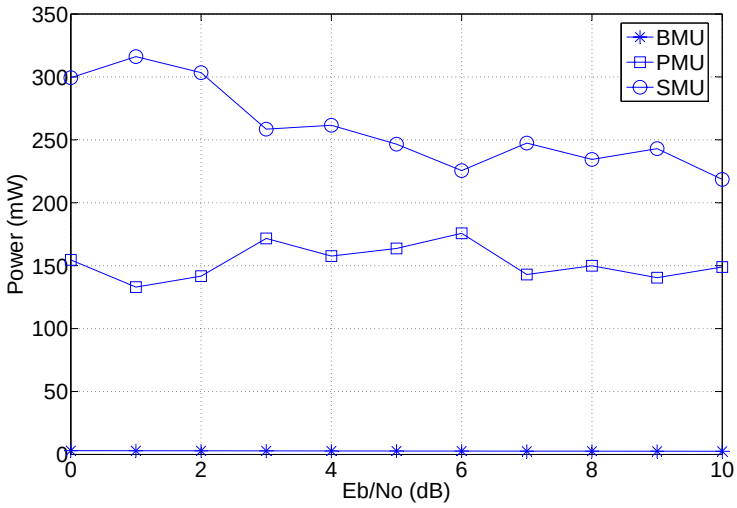
energy efficiency as the constraint length increases, since the energy per corrected bit increases exponentially with the constraint length. However, compared to the increase of power consumption, the increase of energy per corrected bit is less significant, i.e. with the constraint length 8, the energy per corrected bit is increased by a factor of 14 over constraint length 3.

Block power

The standard Viterbi decoder of a constraint length 7 is used to measure the power consumed by the BMU, PMU and SMU at different Eb/No levels. Figure 3.15 shows the results. The power consumption of the Viterbi decoder is dominated by the consumptions of the PMU and SMU, which average 36.8% and 62.5% as Figure 3.15(a) shows. The most power consuming block is the SMU. This is due to the complex trace back logic and the memory (implemented as look up tables) it requires. The PMU also consumes a significant amount of power due to the large number of ACS processors. The BMU, on the other hand, is negligible in terms of power consumption. Therefore, a low power design of a Viterbi decoder should target reducing the power dissipated in



(a) The average percentage of the blocks power consumption.



(b) The block dynamic power consumption at different noise levels.

Fig. 3.15: The blocks dynamic power consumption of a standard (R=1/2, k=7) Viterbi decoder.

the PMU and SMU.

When the noise level changes, as shown in Figure 3.15(b), the BMU power consumption remains constant. For the PMU, the power consumption tends to slightly increase with the Eb/No between 0dB to 6dB and then decreases between 6dB to 10dB. The increase of the PMU power consumption is caused by the increase of the path metric value. At a high noise level, most of the branch metric values are smaller than 7 (a most-likely 1 or 0); thus, the accumulated path metric values are relatively small. As the noise level decreases, the branch metric values are increased and cause the increase of the computational activities of path metrics in the PMU, thus increasing the power consumption. However, when the Eb/No are higher than 6dB, most of the path metric values reach their maximum and are kept unchanged. The number of switching activities is, therefore, reduced and this results in a decrease of the PMU power consumption. For the SMU, the power consumption reduces gradually which indicates a decrease of trace back activity in the SMU.

3.3 Summary

A standard Viterbi decoder design comprises 3 major blocks. The BMU evaluates the received code words and produces the branch metrics; the PMU accumulates the path metrics and selects the survivor paths. The SMU can be either implemented with the shift register or trace back approaches. However, since the shift register approach is less power efficient, most of the Viterbi decoder designs are now using trace back SMUs.

In a CMOS circuit, power dissipation is dominated by the switching power dissipation. Therefore, the principle of low power design with CMOS technology is to reduce the number of switching activities in the design. In order to measure the BER and the power of the Viterbi decoder, the design is implemented on a FPGA device. The dynamic power consumption of the FPGA implementation indicates the switching power dissipation of the design with CMOS technology.

In order to simulate the AWGN channel, a noise generator is designed based on the Box-Muller algorithm. This is used to test the standard Viterbi decoder. The BER figures are measured by in-circuit FPGA simulation running at 100MHz; whereas the power figures are measured by using the Xilinx Xpower software tool.

The power consumption of the Viterbi decoders with different constraint lengths are measured in the tests. The results indicate that the decoder power consumption reduces gradually with the increase of Eb/No, but exponentially increases with the constraint length. In order to analyse the power efficiency of the decoder with different constraint length at different Eb/No levels, the power figures are averaged by the number of corrected errors in one second. This gives an energy figure of the decoder in correcting each error. This analysis shows that the energy efficiency of a standard Viterbi decoder reduces dramatically with the increase of the Eb/No and constraint length. Based on this analysis, the fundamental efficiency issue of the Viterbi algorithm can be revealed: the Viterbi algorithm is noise independent so that computational effort could be wasted in processing an error free sequence. In the power analysis of the blocks in the decoder, the results indicate that the SMU and PMU consume 62.5% and 36.8% of the decoder power. Therefore, a low power Viterbi decoder design should aim at reducing the power dissipation in the PMU and SMU. This chapter indicates that much power is expended processing an error free data stream. The next chapter considers an adaptive design where the decoder is switched off if the data is error free.

4. LOW POWER ADAPTIVE VITERBI ALGORITHM AND DECODER DESIGN

In the previous chapter, the power analysis has indicated that the major efficiency issue of the standard Viterbi decoder is the error independency. To address this issue, the decoder operations should be made to adapt to the variation of noise strength and error probability of the received sequence.

4.1 T-algorithms and adaptive T-algorithm

Adaptive Viterbi algorithms were introduced with the goal of reducing the average computation and path storage required by the Viterbi algorithm. In a typical adaptive Viterbi algorithm [42] of constraint length k, instead of computing and retaining all 2^{k-1} possible paths, only those which satisfy certain path distance conditions are retained at each stage. This is also known as T-algorithm and consists of two major process [42]:

1. A threshold T indicates that a path is retained if its path distance is less than $d_m + T$, where d_m is the minimum distance among all surviving paths in the previous trellis stage;

2. The total number of survivor paths per trellis stage is limited to a fixed number, N_{max}, which is pre-set prior to the start of communication.

The first criterion allows high-distance paths that likely do not represent the transmitted data to be eliminated from consideration early in the decoding process. In the case of many paths with similar cost, the second criterion restricts the number of paths to N_{max}. Careful calculation of T and N_{max} is the key to effective use of the T-algorithm.

If threshold T is set to a small value, the average number of paths retained at each trellis stage will be reduced. This can result in an increased BER since the decision on the most likely path has to be taken from a reduced number of possible paths. Alternately, if a large value of T is selected, the average number of survivor paths increases and results in a reduced BER. However, the increased decoding accuracy comes at the expense of additional computation and a larger path storage memory. The maximum survivor paths, N_{max}, has a similar effect on BER as T. As a result, an optimal value for T and N_{max} should be chosen so that the BER is within allowable limits while minimizing decoding complexity. Several power-sensitive implementations of adaptive Viterbi algorithm architecture have been proposed [43] [44]. In [43], a high-level architectural model of an adaptive Viterbi decoder is described. The threshold T and truncation length of the decoder is varied based on the desired BER, SNR (signal to noise ratio), and code transmission rate. In [44], a systolic architecture with a strongly-connected trellis is used. This architecture provides storage for up to 2^{k-1} paths, but only calculates and stores paths whose costs meet threshold T. Power savings are achieved through reduced storage and computation.

Although there are differences between these implementations, they both have the following problems:

1. The T or N_{max} values are determined by BER or SNR thresholds; they are not fully adaptable for small BER or SNR variations.

2. These algorithms do not take into account the received error patterns. Although the BER or SNR may be the same, the effort needed for correcting errors could be different depending on the characteristics of the errors.

3. All these algorithms yield performance degradations, especially at low SNR levels [45]. The optimum T or N_{max} is determined by simulation which may not applicable to the operating conditions of a real implementation.

Therefore, two fundamental research questions are raised. The first question is on what the adaptive capability of the decoding algorithm should be based? This requires finding a simple but efficient method to indicate an error and error probability of the received code words. The second question is how to make the decoding algorithm adaptive to the error variations; this involves finding the optimum decoding approach in terms of the computing effort and decoding accuracy according to the error probabilities. The answer to the second question depends on the first question, since the variation of the decoding effort is based on the indication of the error probability which is the solution to the first question.

4.2 A new adaptive Viterbi algorithm

An ideal approach to solve the research questions revealed in the last section is to apply the appropriate level of computation so that the effort is just enough for correcting the errors, while avoiding any computation when the received sequence contains no error. This requires the knowledge of the error probability of the received sequence prior to decoding. Although there are other decoding algorithms, such as the soft-output Viterbi algorithm (SOVA) or the maximum a posteriori algorithm (MAP), available which can provide an estimation of the error probability of the received data, they are even more complex than the standard Viterbi algorithm and are not suitable for this task. To decide the proper effort in decoding, a simple method of identifying the error sequence is required.

4.2.1 Error pattern in Viterbi algorithm

As has been discussed in the second Chapter, the Viterbi algorithm maximises the likelihood of the received information in terms of code words path however this may not correspond to the true maximum likelihood of the data. Therefore, once an error event occurs in the code words, a wrong path of length L is chosen and this could

cause a sequence of several random errors in the subsequent decoded data. This is shown in Figure 4.1. The first graph in Figure 4.1 shows the errors in the received

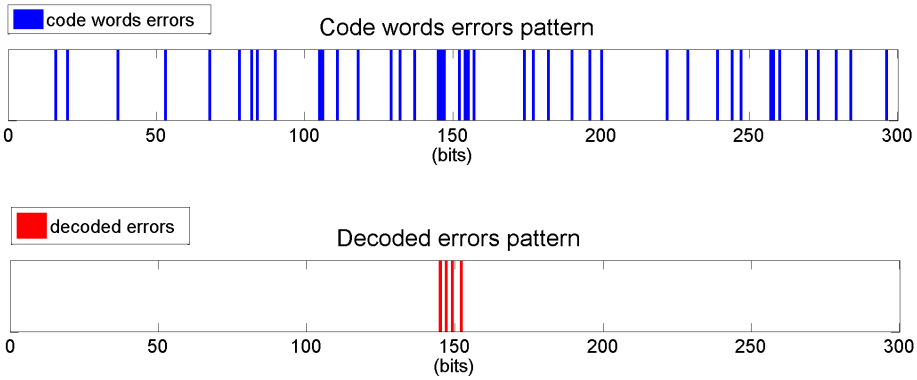


Fig. 4.1: The error pattern in the code words and the corresponding decoded data of a R=1/2, k=7 Viterbi decoder at Eb/No=3dB.

coded words sequence whereas the second graph illustrates the errors in the decoded sequence when this code words sequence is decoded by a Viterbi decoder. Before the first 150 code words, the errors are all corrected by the decoder; however, around the $150th$ code word, errors with a higher density occur in the code words sequence which cause an error event in the code words path. Thus, the wrong path leads to 4 random errors in the decoded data after the $150th$ bit, as shown in Figure 4.1. Two features are suggested by these error patterns correlation between a code words sequence and the decoded data sequence:

1. Firstly, the code words path can be partitioned into subsections with or without errors. Therefore, the Viterbi algorithm is not needed for decoding the sections with no error in the received code words sequence;

2. Secondly, since the decoding process seeks for the maximum likelihood in terms of code words path, if an error event occurs so a wrong path is selected with the Viterbi decoding process, then this results in a random error sequence in the decoded data. In this case, the Viterbi algorithm is not able to improve the accuracy of the decoded data and is thus also not necessary.

Based on these facts, one approach to adaptively decoding the convolutional code with the Viterbi algorithm is to properly identify these two situation and avoid of using Viterbi decoding algorithm. This can be achieved with a simple approach.

4.2.2 No-error code words path identification

As was revealed, to improve its power efficiency a Viterbi decoder should be adaptive to the error variation in the received code words sequence, so that the decoder can be switched off when there is no error in it. Therefore, a simple method of identifying the error free code words sequence is required.

Before discussing the method of identifying a no-error code words sequence, the term of "no-error" code words sequence needs to be defined. In fact, at the receiver side it is not possible to tell if a code words sequence has an error or not. Instead, however, one can predict if a code words sequence will be chosen by the decoder or not based on the decoding algorithm. Thus, in this context, a "no-error" code words sequence refers to the code words sequence which has a zero Hamming distance to an encoded sequence with the same generator polynomial as in the convolutional encoder at the transmitter side of the channel. Therefore, this received code words path will be definitely chosen as the survivor path and used to generate the decoded data by the decoder. In this case, the errors are invisible to the decoder and this sequence, in fact, is treated as "no-error" by the decoder.

Consider the coded digital communication system model shown in Figure 4.2. Let the input and the output of the (n, m, k) convolutional encoder of rate n/m and constraint length k be represented by the n-component vector:

$$X_t = (x_t^{(0)}, x_t^{(1)}, x_t^{(2)}, \cdots, x_t^{(n-1)}) \tag{4.1}$$

and the m-component vector:

$$Y_t = (y_t^{(0)}, y_t^{(1)}, y_t^{(2)}, \cdots, y_t^{(m-1)}) \tag{4.2}$$

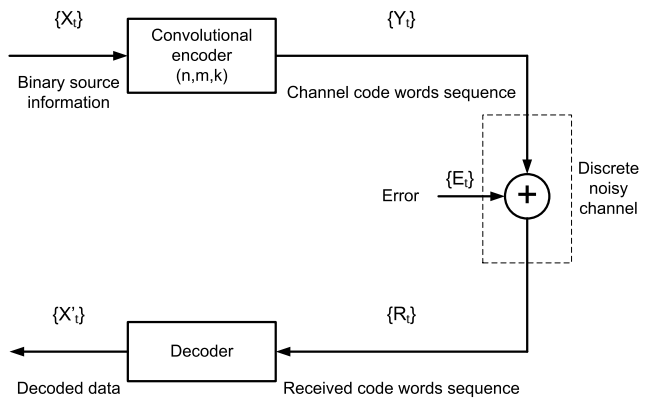


Fig. 4.2: Simple convolutional coded digital communication system model.

for $t \geq 0$, respectively. A binary information sequence $\{X_t\}$ is encoded into a code word sequence $\{Y_t\}$ and sent to a discrete noisy channel at time t. Channel noise cause errors $\{E_t\}$ to be added to the transmitted data. At the receiver, the received sequence $\{R_t\}$ is input to the decoder to recover the information sequence $\{X_t\}$, where:

$$R_t = (r_t^{(0)}, r_t^{(1)}, r_t^{(2)}, \cdots, r_t^{(m-1)}) \tag{4.3}$$

In the Viterbi decoder, the decoded data are recovered from the estimated path $\{\hat{Y}_t\}$ that has the smallest path metric to the received code words sequence $\{R_t\}$ at time t, where $\hat{Y}_t$ can be defined as:

$$\hat{Y}_t = (\hat{y}_t^{(0)}, \hat{y}_t^{(1)}, \hat{y}_t^{(2)}, \cdots, \hat{y}_t^{(m)}) \tag{4.4}$$

From this the decoded output $\{X'_t\}$ is generated.

In hard-decision, as discussed before, the branch metric of estimated code word $\hat{Y}_t$ is the Hamming distance between received code word R_t and $\hat{Y}_t$ at time t which can be denoted as $d(R_t/\hat{Y}_t)$. The path Hamming distance, $d(\{R_t\}/\{\hat{Y}_t\})$ between received code word sequence $\{R_t\}$ and estimated sequence $\{\hat{Y}_t\}$ is simply the sum of the branch Hamming distance from time 0 to t:

$$d(\{R_t\}/\{\hat{Y}_t\}) = \sum_{i=0}^{t} d(R_i/\hat{Y}_i) \tag{4.5}$$

The optimum hard-decision Viterbi decoder determines the code sequence $\{\hat{Y}_t\}$ that is closest in Hamming distance to the received sequence $\{R_t\}$ at time t as the most-likely path. When the received sequence is error-free, $\{R_t\}$ and $\{Y_t\}$ are identical and have the minimum zero Hamming distance over all other possible code word sequences. Therefore, the estimated path $\{\hat{Y}_t\}$ is identical to $\{Y_t\}$ in order to have minimum Hamming distance to $\{R_t\}$. On the other hand, if a code words path has a zero Hamming distance, it will always be chosen by the Viterbi decoder and, thus, treated as "no-error".

In the Viterbi decoding process, the estimated path is identified by tracing back through the path history. In [7], a length equal to 5 times the constraint length is suggested as the minimum trace back depth needed to decode 1/2 codes. This implies that for 1/2 rate, constraint length 7 decoding, if a zero Hamming distance code words path $\{\hat{Y}_t\}$ of length 5 times the constraint length is identified, it will definitely be taken by the Viterbi decoder at time t. The decoder, therefore, can output one bit of decoded data based on the estimated code word $\hat{Y}_{t-35}$ at previous time slot $t-35$, as most of the paths have merged at this point during the trace back process.

The simple convolutional inverse circuit introduced in [28] can be used to pre-decode the data. By re-encoding the pre-decoded data and comparing it with the delayed received code words sequence, the "no-error" path can be easily identified. The architectural block diagram of this new approach, which applies on a rate 1/2 and constraint length 7 code with 3-bit soft decision, is shown in Figure 4.3. The source information sequence is defined by equation (4.6), where X_t is defined in (4.1) with $n = 1$ and is the source information at time t:

$$X = \{X_0, X_1, X_2, \cdots, X_t, \cdots\} \tag{4.6}$$

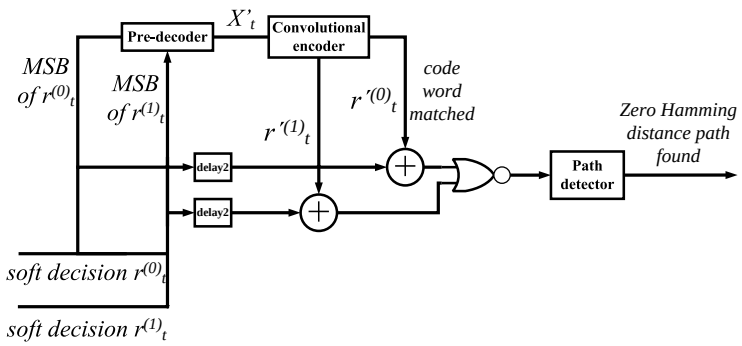


Fig. 4.3: Architecture to identify the zero Hamming distance path for a rate=1/2 and k=7 convolutional codes.

The generator polynomials of the convolutional encoder are expressed as follows, where D denotes the delay operator and "+" denotes modulo-2 addition:

$$G^{(0)} = 1 + D + D^2 + D^3 + D^6 \tag{4.7}$$

$$G^{(1)} = 1 + D^2 + D^3 + D^5 + D^6 \tag{4.8}$$

They are expressed as (171,133) in octal. For a $m = 2$ code, the encoded data at time t, defined as $Y_t = (y_t^{(0)}, y_t^{(1)})$ in (4.2), has two components:

$$y_t^{(0)} = X_t + X_{t-1} + X_{t-2} + X_{t-3} + X_{t-6} \tag{4.9}$$

$$y_t^{(1)} = X_t + X_{t-2} + X_{t-3} + X_{t-5} + X_{t-6} \tag{4.10}$$

By defining hard-decision channel errors at time t as $E_t^{(0)}$ and $E_t^{(1)}$ for $y_t^{(0)}$ and $y_t^{(1)}$, respectively, with the value of "1" when an error occurs, the most significant bits (MSB) of the received code words $R_t = (r_t^{(0)}, r_t^{(1)})$ at time t are expressed by (4.11) and (4.12):

$$r_t^{(0)} = y_t^{(0)} + E_t^{(0)} \tag{4.11}$$

$$r_t^{(1)} = y_t^{(1)} + E_t^{(1)} \tag{4.12}$$

This approach first uses the pre-decoder, shown in Figure 4.3, to pre-decode the source information from R_t. The pre-decoded source information X'_t at time t can be defined by (4.13) [28]:

$$X'_t = (\sum_{i=0}^{4} r_{t-i}^{(0)}) + r_{t-2}^{(1)} + r_{t-4}^{(1)} = X_{t-1} + E'_t \tag{4.13}$$

where

$$E'_t = \sum_{i=0}^{4} E_{t-i}^{(0)} + E_{t-2}^{(1)} + E_{t-4}^{(1)} \tag{4.14}$$

Then, the pre-decoded data is re-encoded by the convolutional encoder with the same generator polynomials $G^{(0)}$ and $G^{(1)}$ as the encoder used for encoding the source information. The re-encoded code words $R'_t = (r_t'^{(0)}, r_t'^{(1)})$ at time t are:

$$r_t'^{(0)} = y_{t-1}^{(0)} + y_{e,t}'^{(0)} \tag{4.15}$$

$$r_t'^{(1)} = y_{t-1}^{(1)} + y_{e,t}'^{(1)} \tag{4.16}$$

where $y_{e,t}'^{(0)}$ and $y_{e,t}'^{(1)}$ are defined by (4.17) and (4.18) representing the convolutionally encoded error sequence $\{E'_t\}$:

$$y_{e,t}'^{(0)} = E'_t + E'_{t-1} + E'_{t-2} + E'_{t-3} + E'_{t-6} \tag{4.17}$$

$$y_{e,t}'^{(1)} = E'_t + E'_{t-2} + E'_{t-3} + E'_{t-5} + E'_{t-6} \tag{4.18}$$

Equations (4.17) and (4.18) from [28] can now be extended beyond. By delaying R_t and if it is identical to the re-encoded data R'_t, then:

$$E_{t-1}^{(0)} = y_{e,t}'^{(0)} \tag{4.19}$$

$$E_{t-1}^{(1)} = y_{e,t}^{\prime(1)} \tag{4.20}$$

Therefore, substituting $E_t^{(0)}$ and $E_t^{(1)}$ in (4.11) and (4.12), yields:

$$r_t^{(0)} = X'_{e,t} + X'_{e,t-1} + X'_{e,t-2} + X'_{e,t-3} + X'_{e,t-6} \tag{4.21}$$

$$r_t^{(1)} = X'_{e,t} + X'_{e,t-2} + X'_{e,t-3} + X'_{e,t-5} + X'_{e,t-6} \tag{4.22}$$

where $X'_{e,t}$ is the modulo-2 addition of the source information X_t at time t and error data E'_{t+1} at time $t+1$:

$$X'_{e,t} = X_t + E'_{t+1} \tag{4.23}$$

The equations (4.21) and (4.22) show that if received code word sequence $\{R_t\}$ equals the re-encoded sequence $\{R'_t\}$, $\{R_t\}$ becomes a convolutionally encoded sequence of $\{X'_{e,t}\}$. Therefore, it is certainly a path having zero Hamming distance to the received code words sequence. The path detector block in Figure 4.3 counts the symbol match signal from current time t up to $t-L$. If all of the previous L symbols are identical, the output is set to indicate that a zero Hamming distance path of length L is found.

4.2.3 A new adaptive Viterbi algorithm and decoder design

Based on the indication of a zero Hamming distance path, a new adaptive Viterbi algorithm and decoder can be proposed. The adaptive algorithm stops the Viterbi decoding process when a zero Hamming distance path occurs and is as follow:

1. Pre-decode and re-encode the received code words R_t;
2. Compare the re-encoded code word R'_t with the corresponding received code words R_t; if they match, set flag $f(t)$ to '1', otherwise set $f(t)$ to '0';
3. In the case of $f(t) = 1$, $count_{wrds} = count_{wrds} + 1$, otherwise $count_{wrds} = 0$;
4. If the $count_{wrds} = 5 * L_c$, where L_c is the constraint length of the code, set flag $ZPath(t - 5 * L_c)$ to '1', otherwise set it to '0';

5. If $ZPath(t - 5 * L_c) = 1$ select the pre-decoded data X'_{t-5*L_c} at time $t - 5 * L_c$ as the decoded output, otherwise apply the Viterbi decoding process on the code word R_{t-5*L_c} and select the corresponding output from the Viterbi decoder as the decoded data.

6. repeat steps (1) to (5) to decode all $\{R_t\}$.

The architectural block diagram of this new adaptive Viterbi algorithm applied on a rate 1/2 and constraint length 7 code with 3-bit soft decision is shown in Figure 4.4. The adaptive part of the decoder is identical to the architecture shown in Figure 4.3. As Figure 4.4 shows, once a zero Hamming distance path is found by the path detector,

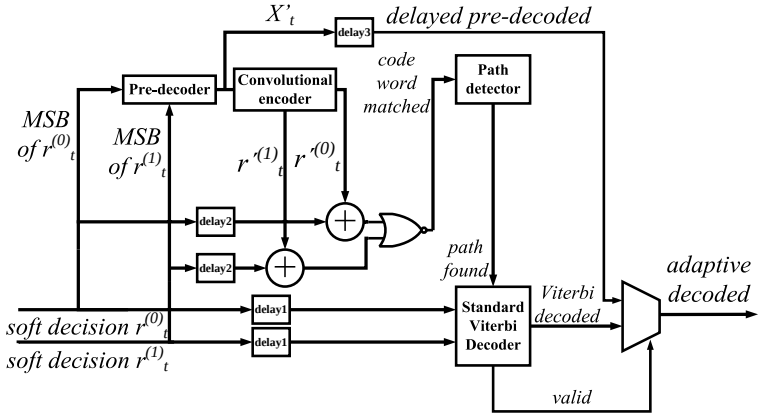


Fig. 4.4: Architecture of the proposed 3-bit soft decision adaptive Viterbi decoder for rate=1/2 and k=7 code.

the Viterbi decoder can be stopped from decoding received code words at time $t - 5 * L_c$ and the pre-decoded data can be selected as the decoded data by the valid signal from the Viterbi decoder.

4.3 BER and power analysis of the proposed adaptive Viterbi decoder

The decoding performance and complexity of the adaptive Viterbi decoder has been tested in Matlab and then implemented on a Field Programmable Gate Array (FPGA).

4.3.1 Matlab Simulation Results

Figure 4.5 shows the bit error rates of the decoded data from the adaptive Viterbi decoder with a zero Hamming distance path of length $L = 35$. In the simulation,

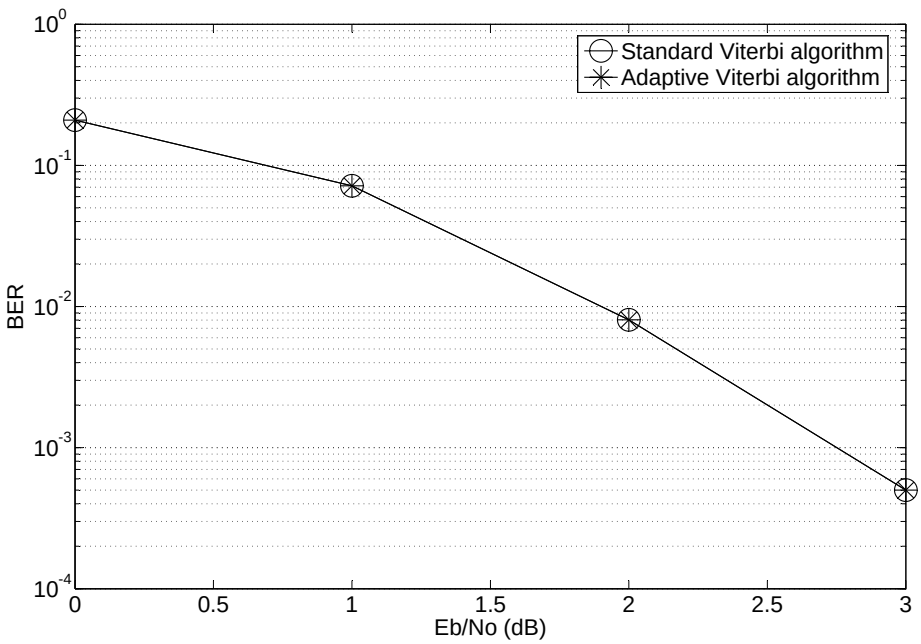


Fig. 4.5: BER performance of the adaptive Viterbi algorithm with $L = 35$, R=1/2, k=7.

all decoded errors are identical to the Matlab conventional Viterbi decoder results and there is no error in the selected pre-decoded data. This suggests the decoding performance of the adaptive Viterbi decoder is comparable to the standard Viterbi decoder.

The percentages of the received code words decoded by the Viterbi decoder or by the pre-decoder are shown in Figure 4.6. As Figure 4.6 shows, when there is a high noise level there is no zero Hamming distance path over 35 time slots long. Therefore, no pre-decoded data is selected by the decoder as the decoded data when Eb/No is less than 3dB; received code words are decoded by the standard Viterbi decoder and there is no power saving. However, when Eb/No is higher than 3dB the number of valid pre-decoded data increases as the noise level reduces. At an Eb/No level of 8dB, the number of Viterbi decoder operations reduces to less than half of the overall number

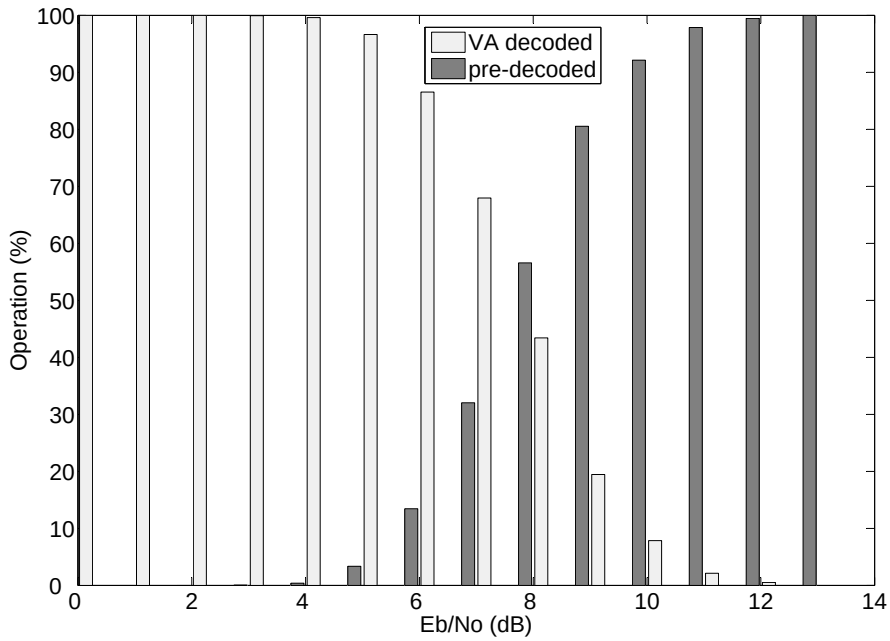


Fig. 4.6: Comparison of pre-decoding and Viterbi decoding operations, R=1/2, k=7.

of decoder code words, which indicates a factor of two reduction in overall Viterbi decoder operations. As Eb/No increases further up to 13dB the pre-decoded circuitry is increasingly used. At 13dB and above, most of the pre-decoded data are valid and Viterbi decoder operations are avoided. At these Eb/No levels, the adaptive Viterbi decoder works mostly as a pre-decoder, and the power consumption is kept very low.

The above results indicate that with $L = 35$, where L is the length of the zero Hamming distance path for the adaptive algorithm, the adaptive Viterbi decoder can save significant power at low Eb/No levels with no accuracy degradation. However, at high noise levels, the reductions are very small. To further increase the saving, L can be reduced so that more zero Hamming distance paths with shorter length can be identified. However, these reductions are not free. As the lengths of the zero Hamming distance paths decreases, they are more likely to be incorrect. Thus, this degrades decoding accuracy.

The adaptive Viterbi decoder has been simulated with different value of L, from 4 to 28 at interval of 4. The BER results are shown in Figure 4.7. The decoder provides

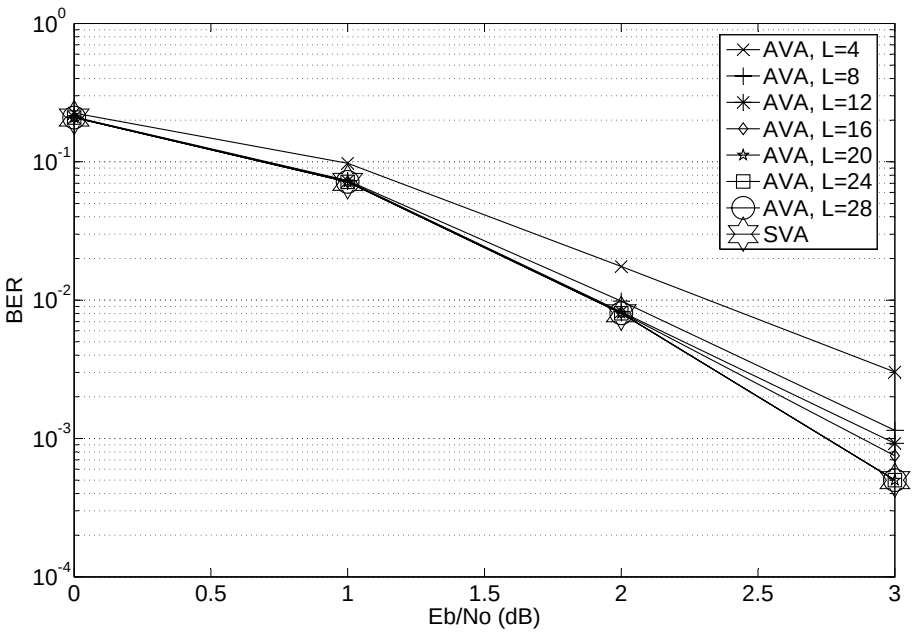


Fig. 4.7: BER performance of adaptive Viterbi algorithm with L from 4 to 28, R=1/2, k=7.

the least decoding accuracy at $L = 4$, and results in an increased BER of nearly a factor of 6 compared to the standard Viterbi decoder at 3dB. The decoding accuracy improves with the increasing L. For L larger than 16, the BER of the adaptive Viterbi decoder is very close to the BER of the standard Viterbi decoder although an increase in the accuracy degradation can be expected at higher Eb/No levels.

With a smaller L, the decoder saves more computations than with a higher L, as shown in Figure 4.8. Lines named with PD in this figure shows that the number of pre-decoding operations increases as the noise level reduces; lines named VD indicate that the Viterbi decoding operations reduces with the noise level. The same saving of Viterbi decoder operations, around 20%, can be achieved at 4dB with $L = 4$ as with $L = 28$ at 6dB, which means further power savings can be made by sacrificing decoding accuracy.

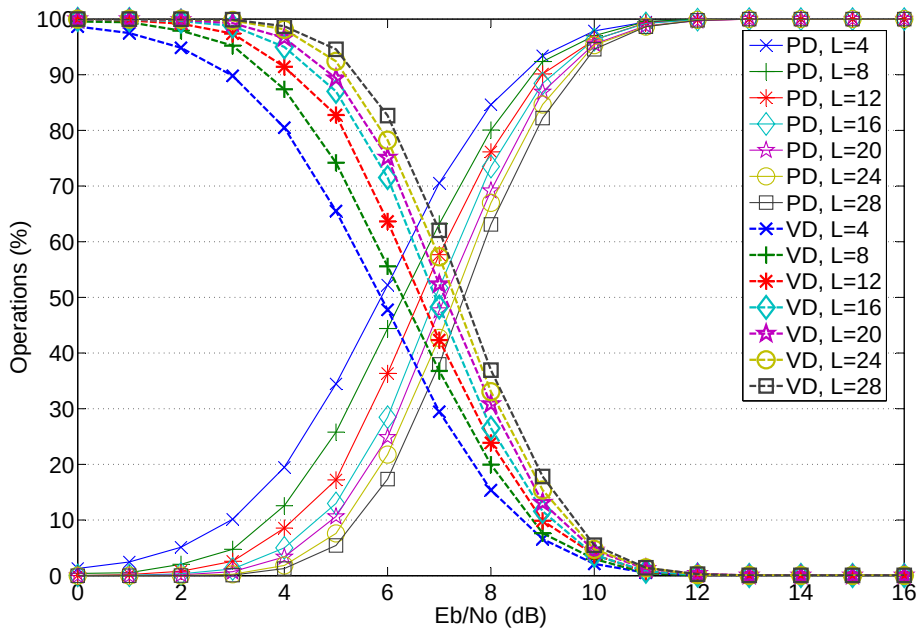


Fig. 4.8: Percentages of pre-decoding and Viterbi decoding operations, R=1/2, k=7.

4.3.2 FPGA simulation results

In order to get more accurate BER figures and evaluate the compatibility of the adaptive algorithm with a standard Viterbi decoder, this new adaptive Viterbi decoder is implemented on a Virtex4 XC4VSX35 FPGA. The post place and route design occupies $2,658$ slices compared with $2,443$ slices of the standard Viterbi core and is 8.8% larger. It runs at a maximum frequency of 165MHz which is the same frequency as the standard one.

Monte Carlo simulations were used to get the BER results shown in Figure 4.9. This figure shows the measured uncoded BER matches the theoretical result very well up to 11dB. This suggests the noise samples are valid in the simulations. Furthermore, the BER from the adaptive Viterbi decoder exactly matches the standard Viterbi decoder BER. Therefore, the adaptive Viterbi decoder provides the same decoding accuracy as the standard Viterbi decoder and yields no accuracy degradation.

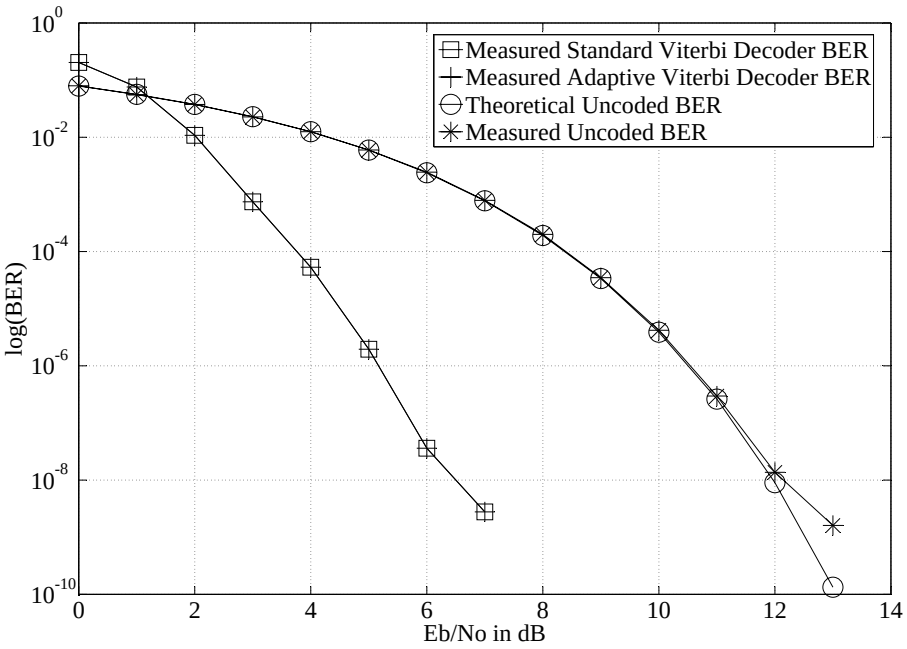


Fig. 4.9: BER performance from FPGA tests, R=1/2, k=7.

4.3.3 Estimated power consumption

With the new adaptive Viterbi algorithm, the whole Viterbi decoder is stopped when the pre-decoded data is valid. Therefore, there is no switching activity in the Viterbi decoder, thus no dynamic power is consumed by the Viterbi decoder. Figure 4.10 shows the dynamic power consumption of this design reduces from 404mW at 6dB to 2.15mW at 13dB. The reduction is from 1.4% to 97%. This indicates that the proposed adaptive Viterbi decoder saves significant power at low noise levels while still providing the same decoding accuracy as a standard Viterbi decoder. However, at higher noise levels, no power can be saved if the same decoding accuracy is maintained. In fact, around 2mW power, which is about 3% of a normal Viterbi decoder power consumption, is consumed by the pre-decoding and this represents the power overhead of this adaptive design.

4.3.4 Comparison of other low power designs

In [43], the proposed adaptive T-algorithm decoder saves 95% of the total energy of a normal Viterbi decoder at 3.75dB. However, it gives a BER of 10^{-4} at this noise

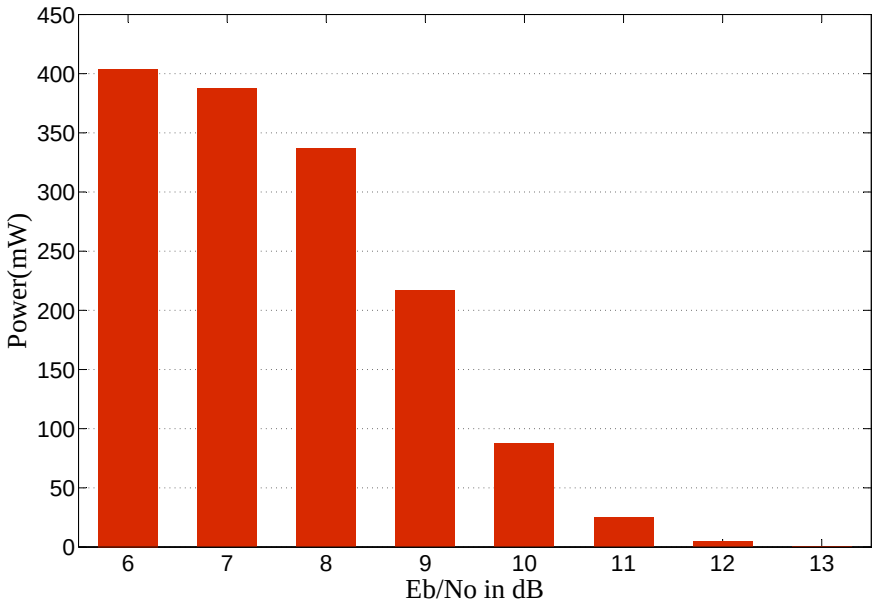


Fig. 4.10: Estimated dynamic power consumption of the adaptive Viterbi decoder on Virtex4 XC4VSX35, R=1/2, k=7.

level. Comparing this to the BER of 3×10^{-6} from a standard Viterbi decoder [7], the adaptive T-algorithm decoder is in fact 33 times worse in decoding accuracy. Therefore, compared with the adaptive T-algorithm decoder, our decoder provides higher decoding accuracy and still achieves significant power saving at low noise levels.

In [28], a scarce-state-transition (SST) system has been proposed. It reports a power reduction of 40% at a BER of 10^{-4} when operating at an information rate of 25 Mbps. The benefit provided by the SST system is that the Viterbi decoder power consumption can be saved without any decoding accuracy reduction. However, because a SST system never stops the Viterbi decoder even when the noise level is very low, the potential power saving is limited compared to our adaptive algorithm, which can save up to 97% overall decoder power at low noise levels.

4.3.5 Possible applications

In practice, the adaptive Viterbi decoder can be used to save power in a system where the BER of the input is lower than 10^{-4}. This is the case in the read channel of

optical storage systems. In a standard DVD player system [21], the BER of the read out signal is between 4×10^{-4} to 4×10^{-6}. To further improve the accuracy of the read out data, the partial response maximum likelihood (PRML) scheme [46] can be implemented with a Viterbi detector. By using a Viterbi detector modified using the proposed adaptive technique instead of a conventional Viterbi detector, up to 73% power can be saved when the BER of the input signal is at 4×10^{-7}.

Furthermore, the adaptive Viterbi decoder will also yield power efficient operation for applications subject to error bursts. Here, the full Viterbi decoding will operate around the error burst with the pre-decoding logic operating in the intervening intervals. In these situations, using the adaptive Viterbi decoder would allow power gains at much lower bit error rates. In a Turbo decoder, either parallel or sequential, two soft-output Viterbi decoders can be used with the proposed adaptive algorithm. Since the output from the first Viterbi decoder tends to be bursty, as indicated in Figure 4.1, the second Viterbi decoder can save significant power with the adaptive approach in each decoding interval.

4.4 Summary

The adaptive approach is one of the major methods for reducing Viterbi decoder power consumption. Designs using this approach, however, are basically trading off the decoding accuracy with the power dissipation by approximating and limiting the path metrics and paths number, such as the adaptive T-algorithm. From a decoding accuracy point of view, this is not efficient since the adaptive capabilities of these approaches are objective and predetermined by the designer so that the decoding processes are not varied subject to the real channel error conditions.

The ideal approach, which has been revealed in this chapter, is to transform the Viterbi decoder from channel error independent to error dependent. More precisely, the decoder should be stopped when there is no error in the received code words and

restarted to correct errors otherwise. To achieve this, a simple method is required to pre-decode and identify the no-error code words sequence. Based on the inverse circuit in [28], a simple approach has been discovered for finding the zero Hamming distance code words path. A new adaptive algorithm, therefore, can be proposed so that the Viterbi decoder can be stopped from processing a zero Hamming distance path and the pre-decoded data from the zero Hamming distance path can be used as the decoded output instead.

Test results show that with the length of a zero Hamming distance path equal to or larger than 5 times the constraint length, the decoding accuracy of the adaptive algorithm is identical to the standard Viterbi algorithm with the same trace back length. Potential reduction of the power consumption in the Viterbi decoder is from 1.4% to 97% as Eb/No increases from 6dB to 13dB. However, the power reduction is small when the noise level is high ($\leq$6dB). Moreover, the ideal of using pre-decoded data from the zero Hamming distance path is decoder independent. Therefore, it should be possible to be adopted in other convolutional decoding applications to minimize power consumption.

5. LOW POWER SMU DESIGN

In chapter 3, it has been revealed that a SMU consumes more than half of a standard Viterbi decoder power. Therefore, a low power Viterbi decoder design can be achieved by minimising the power dissipation of the SMU.

5.1 Design of SMU

Two major approaches, Register Exchange and Trace Back, are commonly used to implement the SMU. As has been discussed in chapter 3, the register exchange approach is not power efficient as a large amount of power is wasted by moving data from one register to another. On the other hand, the trace back approach is generally recognised as the lower power alternative to the register exchange method. Therefore, for a low power Viterbi decoder design, the trace back approach should be adopted.

The trace back process is fundamentally a recursive updating process. This process requires the local winner decisions to be stored in a decision memory prior to tracing back the survivor path. The trace back recursion estimates the previous encoder state S_{n-1} according to the current state S_n as

$$S_{n-1} = S_n[m-2:0]d_n^S \tag{5.1}$$

where $m = (k-1)$ and is the total bits of the state index S_n. For the common radix-2 trellis, the one bit local winner decision d_n^S is read from the local winner memory located by the state index S_n and time index n; the previous state S_{n-1} is obtained by simply discarding the most significant bit of S_n and appending d_n^S as the least significant bit.

5.1.1 Major SMU operations

In the SMU, operations can be divided into two major domains: the updating of new local winner decisions and the trace back decoding operations. The operations of updating the new local winner involve regularly writing the new branch selections into the memory. Since the local winners are produced by the PMU synchronously, the updating operations should also synchronise to the PMU operations. The trace back operations in the SMU can be started at any state. The recursion shown in equation 5.1 is continuously repeated for at least $5k$ times (to avoid degradation in the BER performance) during the trace back process so that the converged path can be reached. Based on the rest of the converged path, the decoded data can be generated. Since trace backs in the SMU are based on the stored local winner they are independent of the memory updating process. Therefore, trace back operations can be performed asynchronously if desired. Currently SMU designs are implemented with full synchronous or asynchronous timing techniques.

Synchronised SMU timing and design

In a synchronised SMU design the trace back is performed discretely. Therefore, during each cycle of updating the new path metric, a trace back can only be performed in a certain number of stages synchronously.

The most common synchronous trace back SMU design is the one-pointer architecture [47]. Figure 5.1 shows the memory architecture of the one-pointer SMU design. The cyclic local winner memory is partitioned into a write block of length D, a merge block of length L and a read block of length D, as shown in Figure 5.1. In the decoding process, the new local winner from the add-compare-select operations in the PMU are written to the write region while the previous local winner decisions are traced and read from the merge and read block, respectively. Once the data is read and decoded from the read block, it becomes the write block for the next round of the decoding process and the other block partitions are shifted accordingly [48]. During each memory

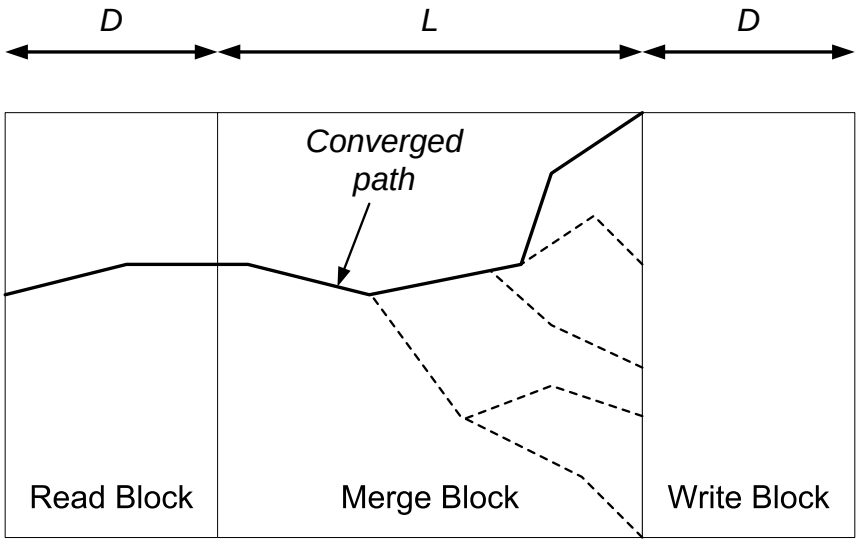


Fig. 5.1: Memory architecture of the one-pointer trace back SMU.

writing cycle one slice of the write block memory is updated by the new local winner information while $(1 + \frac{L}{D})$ cycles of trace back recursions are performed in the read and merge block. Since the clock for the trace back operations is synchronised to the memory write clock and $(1 + \frac{L}{D})$ times faster, a trace back is finished as soon as the write block is fully updated. The SMU can thus operate without any break. For the synchronous trace back approach, the memory length D of the write block depends on the trace back and the memory updating clocks frequencies, f_{tb} and f_w, as

$$D = \frac{L}{f_{tb}/f_w - 1} \quad for\, f_{tb}/f_w > 1. \tag{5.2}$$

Although this synchronised one-pointer trace back architecture is simple to implement, it requires a large memory due to the practical low trace back frequency (typically 2 to 3) caused by the delay of each trace back recursion [48]. Moreover, for the one-pointer approach, if a trace back follows a wrong survivor path at high noise level, a block of incorrect data will be decoded and this can not be corrected by the next valid trace back. This, thus, limits the decoding accuracy of the SMU. The classical solution to

this problem is to run multiple trace back processes concurrently which results in the so called k-pointer trace back architecture [48] [49]. Generally, increasing the number of trace back pointers reduces the required time interval for tracing back and reusing the memory. Thus, it reduces the size of the required local winner memory [49] and the time for correcting a wrong trace back; it thus increases the SMU decoding accuracy. However, due to the overhead of the multiple pointer control logic, the reduction of the design size and power consumption is limited. In fact, with multiple pointers, the memory access rate is significantly increased thus an increase of SMU power dissipation can be expected.

5.1.2 Asynchronised SMU timing and design

On the contrary to a synchronised design, the timing is individually scheduled in an asynchronous system and there is no global timing reference for the state transitions. In an asynchronous SMU design, the trace back is performed continuously controlled by handshakes. Figure 5.2 illustrate the self-timed SMU architecture from [1] which uses a four-phase bundled-data interface [50]. In this design, each trace back is started by issuing an evaluate signal in Figure 5.2, which is then propagated asynchronously through the control logic stages and enables the updating of the (k-1) bits local winner address in each control block. One significant feature of this design is that it only starts a trace back when there is a survivor path with a unique minimum path metric, which is referred to as the global winner in [1]. Moreover, a trace back is forced to stop when the path is merged so the repeated trace back operations as in the synchronised design can be avoided. Since the path reconstruction is only undertaken if necessary and for only as long as required, significant SMU operations are avoided, and this reduces its power dissipation. In fact, with this asynchronous approach, the SMU can have multiple trace back pointers as many as the number of the total memory length. Therefore, the memory size can be minimised to be as small as only five times the constraint length. This is impossible to achieve in synchronised designs. In spite of

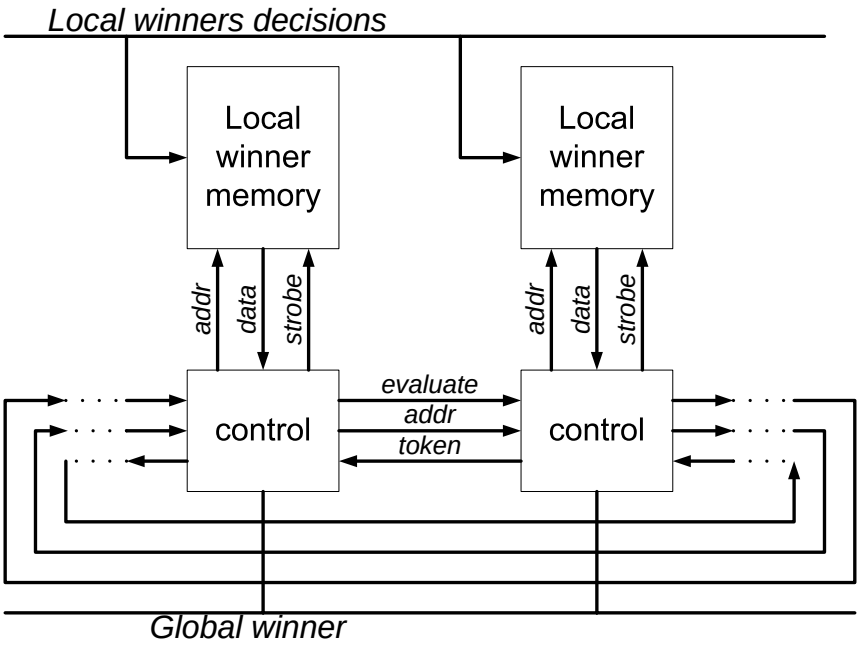


Fig. 5.2: The SMU architecture of the asynchronous design from [1].

these advantages, this asynchronous design has the overhead of handshake logic that consumes extra power due to the trace back handshakes applied at each trace back stage.

In the asynchronous design, the local winner memory is sourced from the PMU and the timing of the memory updating depends on the handshakes between the PMU and the SMU. However, the trace backs in the SMU are controlled by the handshakes between control logic stages and are independent of the handshakes of the memory updating operations. Since the data from each control stage, e.g. the addr and strobe shown in Figure 5.2, and the local winner decisions from the PMU share the same local winner memory, an error will occur when the trace back and the updating operations are accessing the same local winner memory slice at the same time. In this design a trace back has to be forcibly stopped if it is in danger of running into the head slot where the trace back is started. This is resolved by arbiters. Any metastability occurs internally to the arbiters with the outputs indicating '0' until such time as the arbiter resolves the inputs. The arbiters add additional delay to the trace back path.

5.2 New Trace Back SMU design

The advantages indicated by the asynchronous SMU design revealed the fact that trace back operations are more efficient implemented asynchronously. However, the drawbacks of applying only asynchronous or synchronous technique to the SMU design indicates it is ideal to achieve the design efficiency by implementing memory updating operations synchronously while adopting asynchronous timing in the trace back process.

5.2.1 Timing feature of the trace back convergence

In general, the timing control in a VLSI design, either synchronous or asynchronous, is used to guarantee the validity of the output data from a subsystem. In other words, if the output data is always valid, the timing control can actually be avoided and this applies to the trace back implementation. In the trellis diagram of Figure 5.3, the

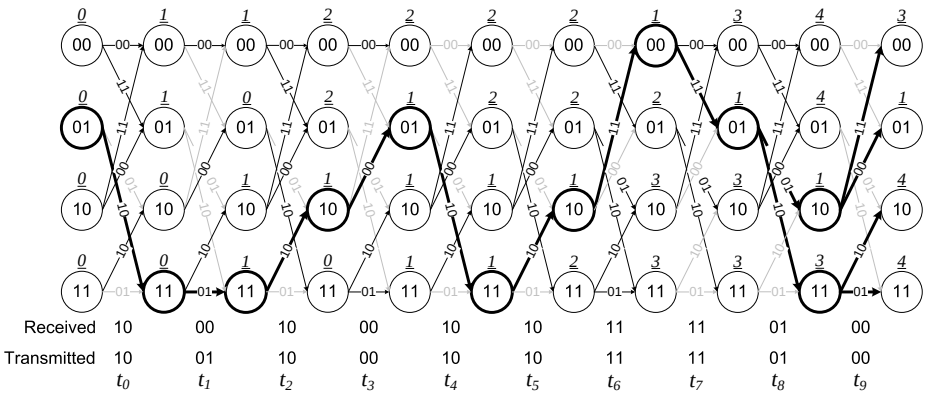


Fig. 5.3: The trellis structure shows the Viterbi decoding process of a R=1/2, k=3 code.

survivor paths from all states, shown as the highlighted lines, at the end of the trellis converge into a single path at t_7. Therefore, as the trellis extends as the time progresses, the converged path will also extend and the previous section of the converged path remains unchanged. Therefore, from the implementation point of view, once the merged

path has been generated, it can be read out at any time regardless of the trace back timing. In this case, trace backs are only required to continuously hold and extend the converged path. This is a very important timing feature of the trace back algorithm. Taking advantage of this feature, the timing control of a SMU design can be simplified.

5.2.2 Overview of the new SMU architecture

According to the timing feature of the SMU trace backs, an efficient mixed synchronous and asynchronous SMU architecture can be proposed. Figure 5.4 illustrates the new SMU top level architecture which is targeted at a Viterbi decoder for decoding 1/2 rate convolutional codes with a constraint length of 7. To ensure the decoding accuracy,

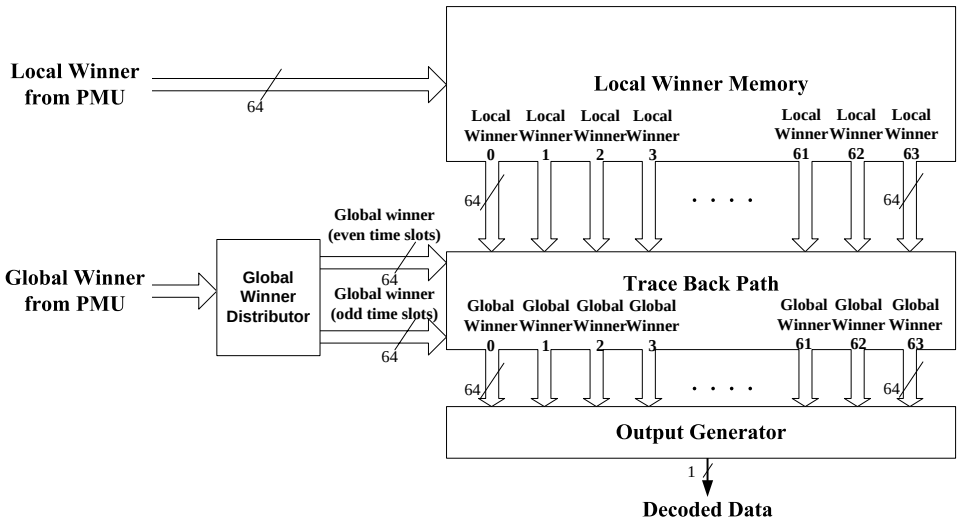


Fig. 5.4: The new R=1/2, k=7, 64-state *SMU* architecture, which consists of four major blocks.

64 trace back stages are used in this SMU design. As shown in Figure 5.4, the design consists of four major blocks: the Local Winner Memory, the Trace Back Path, the Global Winner Distributor and the Output Generator. The Local Winner Memory, the Global Winner Distributor and the Output Generator are synchronised to the global clock whereas the Trace Back Path implements mixed synchronous and asynchronous timing. The local winner memory block is implemented as 64 slots of 64-bit latches and

is loaded synchronously from the system clock. The stored local winner decisions are output straightaway and directly after being updated. This avoids the repeated memory read operations usually performed. Global winners from the PMU are distributed by the Global Winner Distributor block so that the global winners at even and odd time slots are sent to the Trace Back Path block on two different buses. The Trace Back Path is a straightforward implementation of the radix-2 trellis and generates the merged path by trace backs. The global winners from the merged path are produced by combinational logic based on the local winner decisions and the new global winners from the Global Winner Distributor. The Output Generator simply selects the global winner from the converged path and synchronises the decoded data with a flip-flop according to the global clock.

5.2.3 64-bit global winner encoding

In a normal SMU design, the global winner is encoded into $(k-1)$ bits, as suggested in equation 5.1, so that it can address all 2^{k-1} local winners in the memory. This approach requires a decoding process in the memory to transform the global winner index into a single bit read enable signal to access the target location. Also, to produce the previous state S_{n-1}, as indicated in equation 5.1, S_n needs to be shifted and appended by the read out local winner. Since these operations add extra stages to the trace back operations, they introduce delay and timing uncertainty into the design. This, thus, will cause glitches in the trace back process. As discussed before, trace backs are required to constantly hold the converged path without any breaks so that the timing control can be simplified. These glitches in a trace back operation, therefore, must be avoided.

To avoid this, more bits are used so that a global winner is encoded into 2^{k-1} bits with each bit representing whether a global winner appears at each state in the trellis. Therefore, the global winner appearance of a previous state can be derived from the global winner bit of current state and its corresponding bit of the local winner from the

current state; this is computed for each state. With this approach, the global winner encoding and decoding processes are avoided so that the timing control of the trace back process can be largely simplified. Based on the 64-bit global winner encoding, the trace back process can be implemented in a form of the trellis structure.

5.2.4 New trace back path architecture

The trace back path comprises 64 trace back units and is shown in Figure 5.5. Each

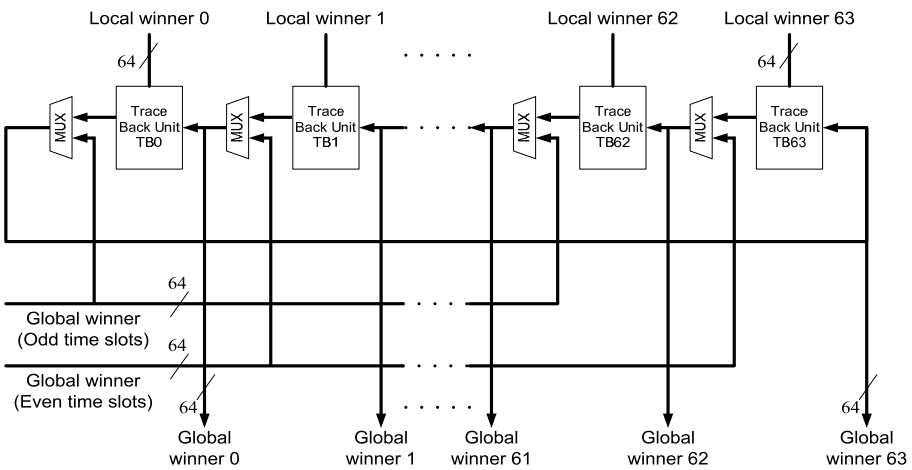


Fig. 5.5: Trace back path of the new *SMU* design.

trace back stage consists of a trace back unit and a multiplexer which selects the global winner input from either the Global Winner Distributor or its predecessor trace back unit. Each stage of the trace back path consists of both synchronous and asynchronous timing.

1. **Synchronised timing in the trace back path.** The selection signal for each multiplexer is synchronised to the updating of the local winner memory and the global winner data from the Global Winner Distributor. When the selection signal is high, the multiplexer of trace back stage n selects the new global winner from the Global Winner Distributor so that the output of the multiplexer is

used by the child trace back stage, stage $(n-1)$, to produce its output global winner; When the selection signal returns low, the multiplexer then selects the output from the trace back unit so that the global winner data produced by current trace back unit can be sent to the child stages and propagated through the rest of the trace back path as a new trace back. The multiplexers in the trace back path work as both the starting points and the end points for new and old trace backs so that the new trace backs are started and the old trace backs are both stopped synchronously by these multiplexers. Although glitches may occur due to the transition of the selection signal, with the overlapped global winner updating scheme used the output of a trace back unit has two clock cycles to settle down before it is selected by the multiplexer. A trace back can thus be started without glitches. Moreover, as normal combinational multiplexers are used, the synchronous and asynchronous data can be fully decoupled without causing any metastability problem. As one of the major features of the new SMU design, the multiplexer timing will be discussed later in section 5.2.6.

2. **Asynchronous timing in the trace back path.** Although trace backs are initialised by the synchronised multiplexers, they propagate asynchronously thereafter through the trace back path, as shown in Figure 5.5. One important feature of this new asynchronous trace back approach is that there is no handshake to control the propagation of a trace back. With the direct output from the local winner memory, fixed survivor branch connections are established in each trace back unit which form the survivor paths for back traces. Since the trace backs are started continuously, the converged path can be held without any handshake inbetween the trace back stages.

The multiplexer selection signal and the global winners from the PMU are synchronised to the updating of the local winner memory. However, the global winner from a predecessor trace back unit is asynchronous to this. If the multiplexer selection

were to be switched just as a new output were generated then spurious transitions would be propagated down the trace back chain. To avoid this, global winners from the PMU update even trace back units on even timeslots, and odd trace back units on odd timeslots. This overlapping enables trace backs to be started when the adjacent global winner has become established and so avoids unnecessary switching transition propagation and avoids additional uncertainty in determining the bit to output. The timing of the multiplexer switching is shown in Figure 5.6. As can be seen, the tran-

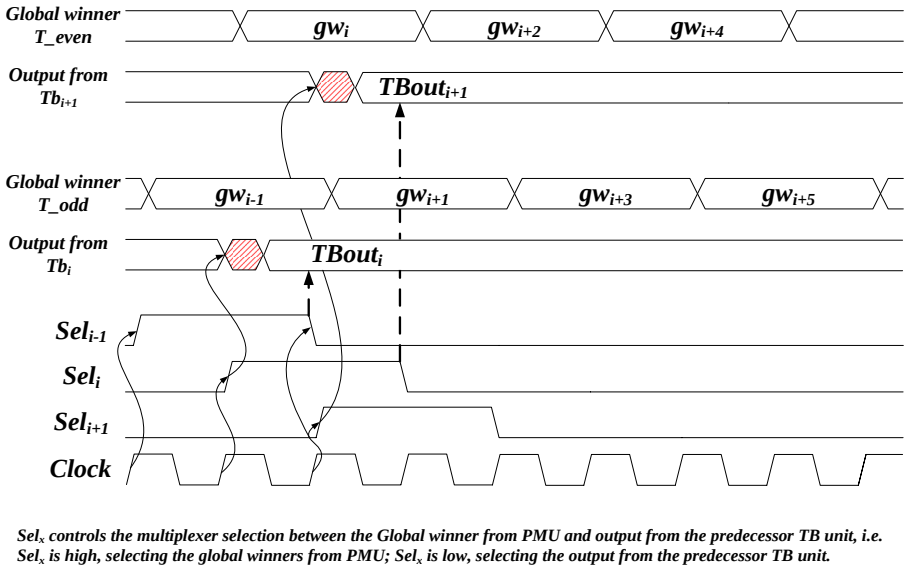


Fig. 5.6: The timing of the trace back path multiplexer.

sitions of Sel signals in Figure 5.6 are synchronised to the clock. In the first clock cycle, the selection signal Sel_{i-1} of the trace back stage $(i-1)$ is set high so that the global winner data at odd time slot *Global winner T_odd* is selected as the output of this stage. At the second clock cycle, the selection of the trace back stage i is set high so that the global winner data at even time slot *Global winner T_even* is selected as the output from the trace back stage i, which is shown as $TBout_i$ in Figure 5.6. Since the selection signal of the trace back stage $(i-1)$ is still high, $TBout_i$ and the possible glitches will not propagate and affect the output of trace back stage $(i-1)$ and the stages thereafter. At the third clock cycle, Sel_{i-1} returns low so that the output from the trace back stage i can be selected and passed through. It then propagates

asynchronously as a new trace back. Since there is one clock period for $TBout_i$ to settle down, glitches caused by the selection signal Sel_i and the trace back logic are prevented from being sent out as a trace back. In this way, the existing converged path can be preserved by continuous trace backs started in an interval of one clock cycle.

In the trace back path, each trace back unit is a direct implementation of one stage of the radix-2 trellis, as shown in Figure 5.7, so that the global winner at time slot T_i is constructed based on the local winner selections and global winner at time slot T_{i+1} in a trace back unit. As shown in Figure 5.7, each bit of the global winner signal

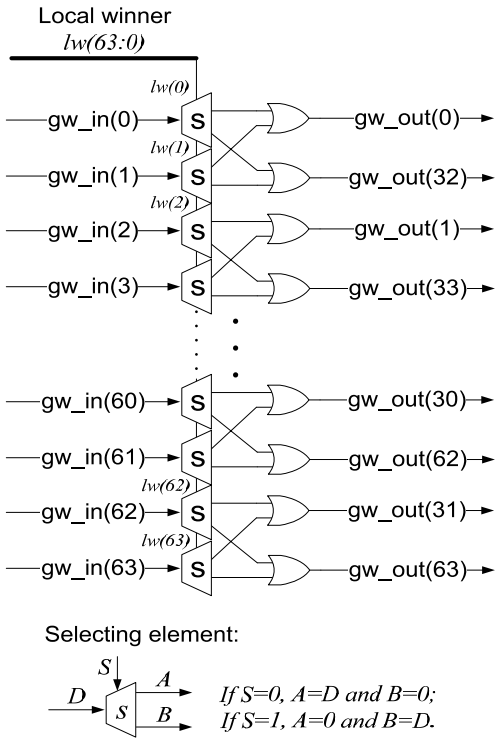


Fig. 5.7: The one stage trellis structure of the Trace back unit.

is diverted by the selecting element (its function is indicated in Fig 5.7) according to the local winner selection of that state. The survivor paths therefore are formed in the trace back path as the established connections through trace back units. The global winner signal from two different states are merged by an OR gate. The OR gate is a simple but efficient way of implementing the path convergence as there is no extra

logic required for detecting the convergence to stop a merged trace back as in the handshaked asynchronous design. With this approach, a trace back propagation stops once it 'OR'ed with an existing trace back signal and no further transition occurs at the OR gate output thus minimising the power dissipated in trace backs.

5.2.5 Local Winner Memory

The local winner memory is used to store the local winners from the PMU. This is a circular buffer with a forward moving token in which one register is needed to record the local winner of each state at every time slot. The moving of the token is synchronised to the PMU clock. For a 64 state, 64 time slots SMU, 64×64-bit registers are required as illustrated in Figure 5.8. Register R_i in Figure 5.8 holds the local winners of all states

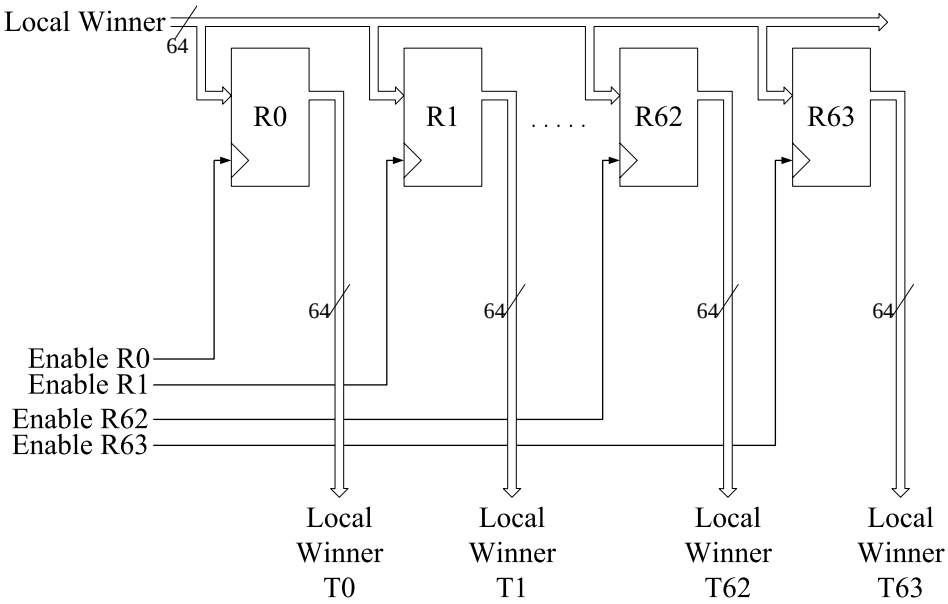


Fig. 5.8: Local winner memory of the new *SMU* design. It uses latch registers to store local winner information.

at time slot i, where i is 0 to 63. Instead of RAM or D-type flip-flops, transparent latches are adopted in order to minimize the power dissipated when writing. Updating of each register is controlled by synchronised enable signals so that the local winners from the PMU fill successive registers as time progresses. This scheme is called selective

update in [51]. When the *ith* data is received, only register R_i is enabled so that the local winner of all the states at time slot i is recorded in the register. Since clocks are disabled for other registers, this scheme can reduce switching activity to a minimal level to save power [51].

Furthermore, instead of having a single 64-bit output, this local winner memory has 64 64-bit outputs, shown as T_i in Figure 5.8. Therefore, there is no need for any read operation compared with conventional SMU memory using RAM storage thus reducing switching activity and unnecessary power dissipation.

5.2.6 Global Winner Distributor

In the Global Winner Distributor, the global winners from the PMU at even and odd time slots are clocked alternately into Register A and Register B using a half-frequency-clock and its inverse, see Figure 5.9; they are then distributed onto two global winner buses, A and B. As shown in the timing of Figure 5.10, a 'current' and 'previous' global

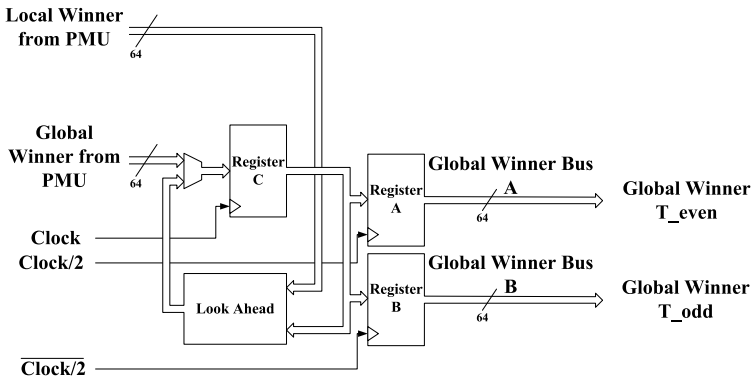


Fig. 5.9: Global Winner Distributor of the new *SMU* design.

winner are held constant over two time slots on the global winner buses A and B. In the Global Winner Distributor, look-ahead logic is used to trace forward and estimate the global winner for time slot T_{i+1} based on the global winner at T_i and the local winner at T_{i+1}. This is only used in the case when the global winner from the PMU is invalid, e.g. no state metric is zero; in this case, the Global Winner Distributor just inserts the estimated global winners onto the global winner buses so that the trace

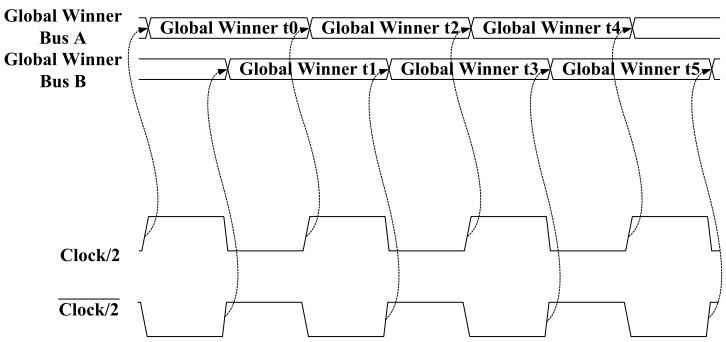


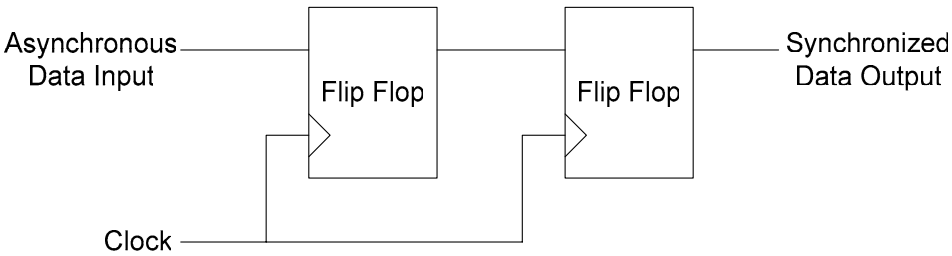
Fig. 5.10: Timing of the global winner buses A and B

backs starting from these states can still extend and hold the existing converged path, avoiding unnecessary transition changes. This is a simple solution which avoids of using complex control logic or memory to keep the existing trace back status. With this new trace back path design, multiple global winners can be injected into the trace back path to start more than one trace backs simultaneously.

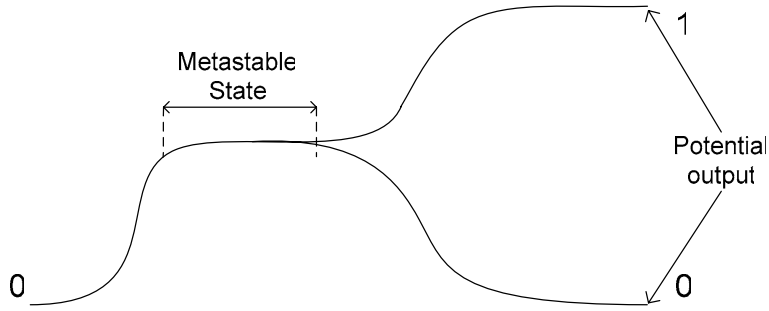
5.2.7 Output Generator

The output generator block simply selects and clocks the global winners of the oldest time slot from the trace back path into a register, just before it is updated by the PMU. The output generator block is synchronised to the global clock; thus, the input and output of the new SMU are matched without using any buffer. The 64-bit global winner information is then decoded into a single bit data with a '0' output if the '1' indicating the global winner is in an even state position and a '1' output for an odd state position. In the case of more than one global winner being indicated at the oldest time slot, as a trace back can start with multiple global winners, the output could either be '0' or '1' depending on the decoding logic design.

In this new SMU architecture, the global winner signals from the trace back path are asynchronous with respect to the synchronizer flip-flop in the output generator. Therefore, when the output generator clocks the global winners into its flip-flop registers and produces the output, it synchronizes the global winners output with the global



(a) Metastable state can be resolved into either '1' or '0'.



(b) A synchronizer using cascaded flip-flops allows one clock period for the data to resolve.

Fig. 5.11: Metastability and a simple flip-flop synchronizer.

clock. Metastability or glitches can thus occur when the global winner is changing at or close to the system clock edge. The new data, therefore, may or may not be entered in the output register due to different flip-flop set up and hold times. Furthermore, flip-flops can enter a metastable state where a non-standard "half-level" is output as shown in Figure 5.11(a). In the new SMU, a cascaded flip-flop synchronizer shown in Figure 5.11(b) is implemented to save the asynchronous global winner information. This allows one clock period for the data to resolve before being output and such synchronizers minimize the effect of metastability. According to [52] [53] [54], the mean time between failure (MTBF) of this synchronizer can be calculated using the equation below [50].

$$MTBF = \frac{e^{t/\tau}}{T_w \times f_d \times f_c} \tag{5.3}$$

where f_d and f_c are the frequency of data and clock transitions; T_w is the time interval between the clock and data transition giving rise to a non-zero resolving time; t is the time allowed for resolving and τ is the time constant for leaving the metastable state.

This equation indicates that the MTBF is exponentially proportional to both t and τ ; thus, increasing t or minimizing τ are preferable in order to increase the MTBF. τ and T_w are estimated using the simple set-reset latch shown in Figure 5.12. By violating the

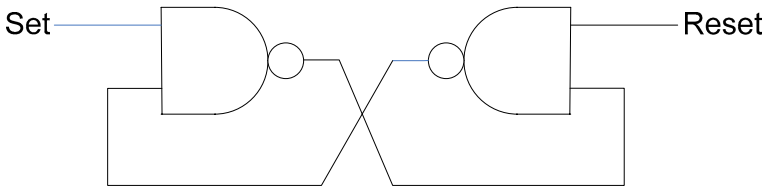


Fig. 5.12: Metastability simulation circuit in measuring τ.

setup and hold time constraints for the set and reset signals, metastability is generated. In the post layout simulation with .18 micron CMOS technology, τ is measured to be from $14ps$ to $43ps$ and T_w is from $16ps$ to $21ps$. Using the maximum τ and T_w, the minimum MTBF of the flip-flop synchronizer is calculated to be 10,209 years for a 45MHz clock frequency and 1,087 years for a 100MHz clock frequency. Therefore, using the flip-flop synchronizer of Figure 5.11(b) is sufficiently reliable. Although the synchronizer may resolve to a random output, this will not affect the decoding accuracy. The trace backs are rarely correct if they can reach the oldest time slot without merging into the converged path. In most of the cases correct trace backs will merge together considerably before reaching the oldest time slots in a frame. If a trace back runs back to the oldest time slot the correct trace back is destroyed and this causes wrong data to be decoded. Therefore, even if there is no synchronisation problem, the decoded data from these long trace backs will still produce wrong output data. So, it is not necessary to add extra logic to avoid the random errors caused by metastability at the oldest time slot. This again simplifies the SMU architecture and minimizes its power consumption.

5.3 Timing in the new SMU design

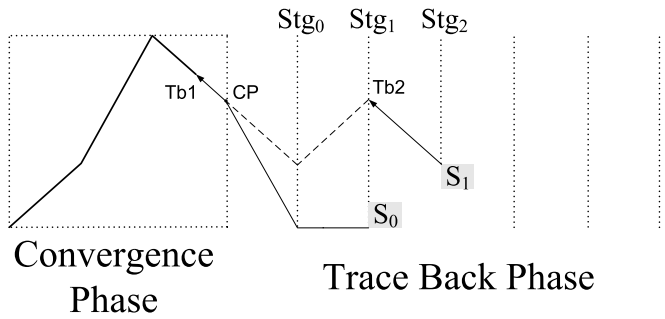
Since there is no handshake to control the trace backs in this new SMU design, timing skew could occur. However, timing skews do not always causes the failure in a design. For a trace back, there are two types of timing skews which may occur in this new SMU design and can be referred to as positive timing skew and negative timing skew.

5.3.1 Positive timing skew

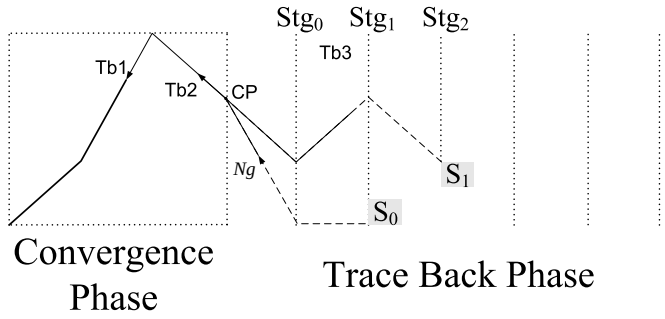
At each time slot, a trace back is started by sending a logic '1' signal from the global winner state. Since the trace backs are travelling asynchronously in the trace back path, the delay may cause the logic '1' signal sent at time t_n to propagate slower than that sent at t_{n+1}. This is referred as the positive timing skew since the new trace back travels faster than the old one.

Figure 5.13(a) and Figure 5.13(b) show the trace back path status in the positive timing skew situation at the time frame t_2 and a short period after t_2, at $t_2 + \Delta$. In Figure 5.13, vertical lines illustrate the trace back stages in the new SMU design. The solid lines in Figure 5.13 indicate the paths having a logic '1' status which represent the global winner states; the dashed lines indicate the path with logic '0', i.e. non-global winner status. At time t_2, as shown in Figure 5.13(a), the logic '1' signal, $Tb2$, sent from Stg_2 has reached Stg_1. At the same time, Stg_1 is still being updated by the PMU and sending the logic '1' signal, shown as $Tb1$, which holds and indicates the converged path so that the decoded data can be generated. However, due to the overlap of updating in the trace back path design, $Tb2$ is not allowed to pass through Stg_1 at t_2.

Immediately after t_2, Stg_1 stops updating and allows the trace back signals to pass through so that they can travel freely on the rest of the trace back path. Since the path start at Stg_1 and Stg_2 is different, the non-global winner states indicated by logic '0' signals at Stg_2 will also pass through Stg_1 simultaneously. These changes alter S_0



(a) Trace back status at time t_2



(b) Trace back status at time $t_2 + \Delta$

Fig. 5.13: Trace back status in positive timing skew situation.

to the non-global winner state so that there is still one global winner state at Stg_1. In the positive timing skew situation, as shown in Figure 5.13(b), since the trace back signals from Stg_2 travel asynchronously without any timing control, the logic '1' signal $Tb2$ is propagating faster than the logic '0' signal Ng due to the different delay in each path. However, because the decoded data is decided by the converged path and the positive timing skew does not affect its existence as $Tb1$ and $Tb2$ merge together at CP, as shown in Figure 5.13(b), there will be no decoding error caused by the positive timing skew.

5.3.2 Negative timing skew

The timing skew illustrated in Figure 5.13 is one type of situation where a logic '1' signal sent at time t travels faster than the one sent at $(t-1)$ so that it can always catch up and merge with former '1's. However, it may occur that the later logic '1' signals may propagate slower than the former so that a gap forms in between. This is referred to as negative timing skew of the trace backs and is illustrated by Figure 5.14. The solid line from state S_1 represents the global winner propagation with time and the

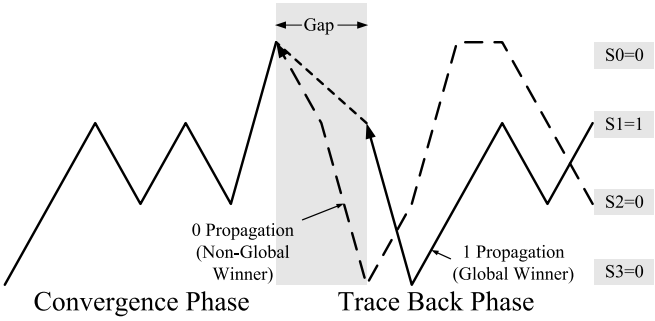
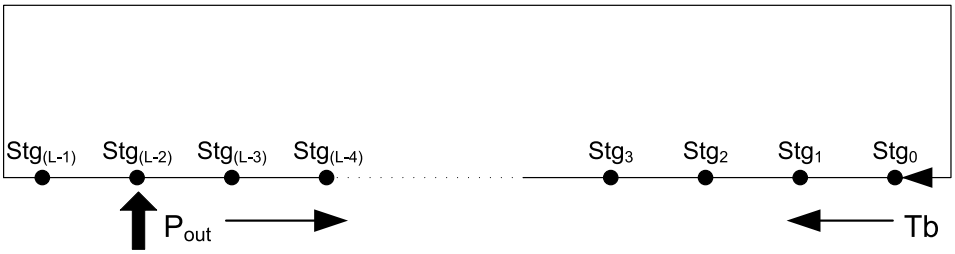


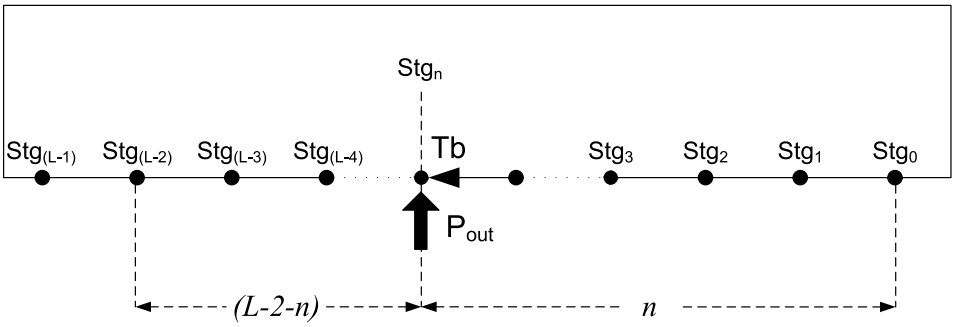
Fig. 5.14: Trace back gap caused by timing skew.

dotted line represents all the other 'loser' states. The slower propagation of the winner means that the loser '0' states can combine with a zero on the winner path from the previous timeslot to indicate a period where no winner is indicated; this is indicated

by the shaded region in Figure 5.14. Were this to happen at the time the decoded data is clocked into the output flip flop then incorrect data would be decoded and output. After loading a timeslot, the load pointer moves on forward and the trace back moves in the reverse direction. Therefore, the trace back path design in the new SMU can be



(a) Trace back status at time $t_{n+1} + \tau$



(b) Trace back status at time $t_{n+2} + \tau$

Fig. 5.15: Trace back status at various time points.

simplified as shown by the trace back model in Figure 5.15. In Figure 5.15(a) the black dots represent the trace back stages; together with the connections between stages they form the circular trace back path implemented in the new SMU. The arrow P_{out} is the pointer for clocking output data and is moving from Stg_{L-2} to Stg_0 where the L is the total number of trace back stages; $L-2$ arises from the timing of the switching of the multiplexers in the trace back stages. Due to the overlapping of the global winner update, the stage $Stg_{(L-1)}$ is updated by the new global winner while the trace back is starting at stage Stg_0. Therefore, the data is taken from the stage $Stg_{(L-2)}$ as shown

in Figure 5.15.

Since the data is taken out periodically based on the clock, P_{out} can be considered moving at a constant speed with a time interval equal to the clock period T between two stages. Therefore, the data is taken out only at the time when P_{out} passes each stage. The arrow Tb at Stg_0 on the trace back path represents the trace back starting from this stage. It moves towards P_{out} from Stg_0 to Stg_{L-1}. Eventually, P_{out} and Tb will meet each other, as shown in Figure 5.15(b), when they both arrive in Stg_n at the same time. The trace back Tb, thus, travels n stages in this time whereas the P_{out} passes $(L-2-n)$ stages as indicated in Figure 5.15(b). If the trace back delay per stage is d and the clock time is T

$$n \times d = (L-2-n) \times T \tag{5.4}$$

It is the range of delay in a trace back unit that gives rise to the 'zero' gap that may arise. To be more realistic and accurate, the flip-flop setup time t_{setup} and a possible tolerance margin should be included which gives

$$n \times d + (t_{setup} + \beta) = (L-2-n) \times T. \tag{5.5}$$

where β is the tolerance of the clock. Based on this equation, the n of the Tb can be obtained as a function

$$n = \frac{(L-2) \times T}{d+T} - \frac{t_{setup} + \beta}{d+T} \tag{5.6}$$

determined by L, d and T, where the t_{setup} is a constant and β depends on T. Post layout measurements for the $0.18\mu m$ 1.8V process targeted reveal a variation between d_{min} and d_{max} of 0.55ns to 0.615ns. The flip-flops setup time is $0.16ns$ according to the manufacturer's data and 20% tolerance is assumed. Taking a SMU path length L as 64 and using equation 5.6, the variation in n at different frequencies can be computed and are as shown in Table 5.1. It can be seen that despite the delay variations to be

expected in the trace back units, the output data is always within the same stage at frequencies 10MHz, 100MHz, 125MHz and 200MHz. Therefore there will be correct output over these frequencies. The results for 50 and 150MHz indicate that the output

Tab. 5.1: Minimum and maximum trace back stages at a $0.18\mu m$ geometry.

Frequency (MHz)	**Min Stages (n_{min})**	**Max Stages (n_{max})**
10	61.42	61.46
50	59.95	60.14
100	58.20	58.56
125	57.37	57.80
150	56.56	57.07
200	55.00	55.65

could fall in different stages and could therefore cause output errors.

5.3.3 Timing of the scaled new trace back design

As geometries scale down, the stage delay d will alter and the negative timing skew may cause the trace backs to fail. According to the first-order 'constant field' MOS scaling theory [37], scaling a process down by a factor α reduces the gate delay by the same factor α while the wire delay remains the same. Based on this, the variation for d_{min} and d_{max} is shown in Table 5.2. Equation 5.6 can be used to find the variation

Tab. 5.2: Minimum and maximum delays of each trace back stage for different geometries.

Geometry (nm)	**Min Delay (ns)**	**Max Delay (ns)**
180	0.550	0.615
130	0.400	0.465
90	0.275	0.340
60	0.183	0.248

in n at these geometries and this is shown for a 50MHz clock and 64 stages ($= L$) in Table 5.3. As the results suggest, the output data is always within the same stage and so will be correctly output at 50MHz over the range of geometries shown. The results for 90nm in Table 5.4 indicate that the output may fall in different stages at 100MHz and could cause an output error. This can be avoided by either shifting the output clock edge or by altering the number of trace back stages, according to the

Tab. 5.3: Minimum and maximum trace back stages at 50MHz and L=64.

Geometry (nm)	**Min Stages (n_{min})**	**Max Stages (n_{max})**
130	60.39	60.58
90	60.76	60.95
60	61.04	61.23

Tab. 5.4: Minimum and maximum trace back stages at 100MHz and L=62.

Geometry (nm)	**Min Stages (n_{min})**	**Max Stages (n_{max})**
130	59.04	59.41
90	***59.75***	***60.13***
60	60.29	60.67

Equation 5.6. It does indicate that to achieve an efficient design without handshakes is not without problems and that careful timing analysis and simulation is required at the working frequencies and on the targeted process to verify correct operation.

5.4 Test results of the new SMU design

The Viterbi decoder design with this new SMU architecture is implemented in both CMOS circuitry and a FPGA. Power simulations are used to estimate the power figures of the CMOS and FPGA implementations. The BER of this new design is obtained by running Monte Carlo simulations with the FPGA implementation.

5.4.1 CMOS implementation results

The new SMU operates at frequencies 45MHz and 100MHz; and uses a 0.18 micron technology and a 1.8V supply voltage. The layout has been automatically generated using a commercial tool from logic schematics comprising elements from an in-house library of conventional CMOS logic circuits. This approach results in random delays in the trace back path. Table 5.5 summarizes its characteristic with Figure 5.16 showing its layout. The design has been tested by running Nanosim post-layout simulations. In the post-layout tests, three different size ($5k$, $10k$ and $50k$) of random data patterns were generated and added with white Gaussian noise according to the signal noise

Tab. 5.5: Characteristics of the new SMU core

Throughput	$45Mbits/s$ and $100Mbits/s$
Rate	1/2 (or punctured 2/3 to 7/8)
Trace back length	64
Core size	$1.05 \times 1.05 = 1.10mm^2$
Transistors	$241K$
Technology	$.18\mu m$ standard cell

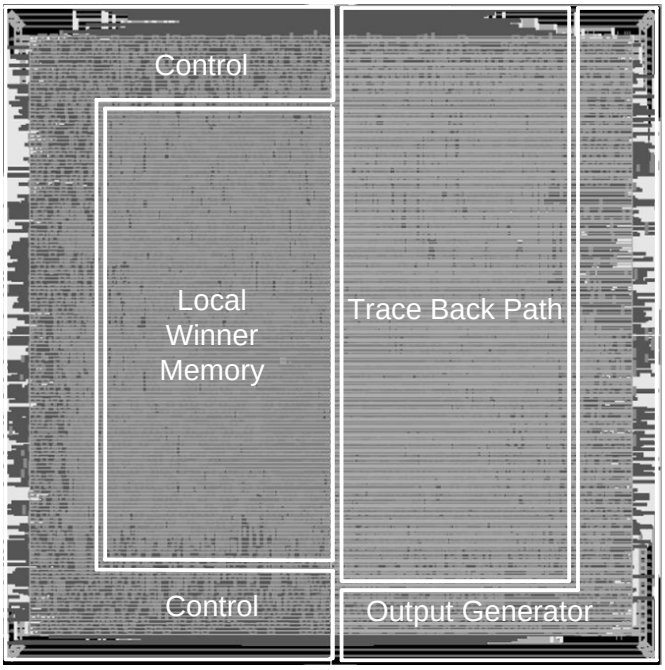


Fig. 5.16: Post-layout of the new SMU core.

ratio in decibels. The resulting output bit error rate (BER) and power consumptions for different signal to noise ratios for code rate 1/2 and punctured 2/3 are given in Table 5.6 and Table 5.7. The output throughput is 45Mbit/sec in these simulations which is equal to the throughput of the reference designs in [27] and [1]. It can be seen that the power increases only relatively slowly with increasing input BER. This suggests that trace backs in the new SMU consume only a small portion of the overall SMU power.

The average power consumption and area of this new SMU design is compared in Table 5.8 with low power Viterbi decoder designs from [27] and [1], which are implemented with single-ended pass-transistor logic (SPL) and asynchronous logic respec-

Tab. 5.6: New SMU BER and power consumption in decoding 1/2 codes

S/N **(dB)**	**Input BER**	**Output BER**	**Power**(mW)
0.4	0.07335	0.01524	6.65
1.7	0.04234	0.00029	6.31
2.6	0.04234	$< 4.85e-5$	6.24
3.5	0.01904	$< 2.31e-5$	6.17

Tab. 5.7: New SMU BER and power consumption in decoding 2/3 codes

S/N **(dB)**	**Input BER**	**Output BER**	**Power**(mW)
1.1	0.05720	0.01350	6.44
2.2	0.03595	0.00080	6.39
3.0	0.02494	$< 6.25e-5$	6.20
3.7	0.01619	$< 2.98e-5$	6.08

tively. Since the SPL design is implemented with the .35 micron CMOS technology and a 3.3V supply voltage its power and area are scaled down by factors of 8 and 4 respectively. The power results have all been scaled (where required) to a 45Mb/s data rate for a .18μm process running from 1.8V.

Tab. 5.8: The proposed SMU power consumption comparing with other low power SMU designs at 1.8V and 180nm.

	New SMU	**SPL [27]**	**Asynch [1]**
Decoder Area(mm^2)	N/A	1.64	1.96
SMU Area(mm^2)	1.10	1.04	~ 0.98
No. of States	64	64	64
Avg. Power(mW) **of Decoder @45Mb/s**	N/A	62.5	~ 18
Avg. Power(mW) **of SMU @45Mb/s**	6.31	39.38	~ 9

According to the comparison, the new SMU architecture only uses 16% of the power dissipated in the low power design using SPL logic. In the asynchronous design from [1], the overall decoder power consumption is also much less than the synchronous SPL design from [27]. Comparing it to the power figure from the asynchronous design, the new SMU architecture can provide a 29.8% power reduction. So, the new SMU is the most power efficient design amongst these three. Also the significant power saving in our new SMU and in the asynchronous design over the synchronous approach

confirms that the asynchronous architecture is definitely beneficial for lowering power dissipation in the SMU implementation. The layout for the SPL and asynchronous SMU designs were done manually while the new SMU layout was generated by the tools. Even so, the areas of all these three implementations are comparable indicating that this proposed architecture is most suitable for trace back SMU implementations which require low power consumption.

5.4.2 FPGA implementation results

Since the post-layout power simulation is extremely compute intensive, it does not provide BERs that are accurate enough at low noise levels. The design has therefore been transferred to a $90nm$ FPGA. Two Viterbi decoders, $R = 1/2$, $k = 7$ and $R = 1/2$, $k = 3$, using this new SMU architecture were implemented on a Virtex4 XC4VSX35 FPGA board. The test framework includes a Viterbi decoder IP core from Xilinx which is compatible with many common standards such as DVB, 3GPP2, IEEE802.16, Hiperlan, and Intelsat IESS-308/309. The power figure is estimated by the Xpower tool from Xilinx. The size of the post place and route design is larger than the Xilinx IP core, as shown in Table 5.9. It shows the Viterbi decoder with the new SMU is 21% and

Tab. 5.9: Number of slices the Viterbi decoder occupied with the new SMU architecture

	RAM Blocks	**Slices**
VD with new SMU(k=3)	0	284
Xilinx IP(k=3)	2	224
VD with new SMU(k=7)	1	3,686
Xilinx IP(k=7)	5	2,423

34% larger than the Xilinx Viterbi decoder of constraint lengths 3 and 7, respectively. All of the designs can operate at a frequency of 100MHz. Monte Carlo simulations were used to get the BER results shown in Figure 5.17. It shows the measured BERs from the Viterbi decoders with the new SMU match the standard Viterbi decoders BERs. There are only small differences at high noise levels. This is due to the extremely small difference between the trace back speeds in the new SMU and the Xilinx Viterbi

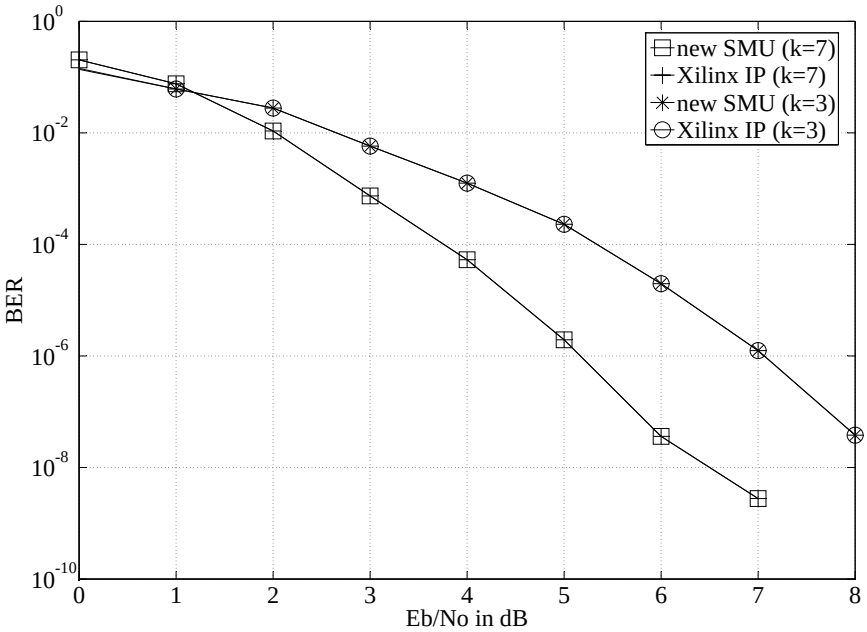


Fig. 5.17: BER performance from FPGA tests.

decoder. This indicates the trace backs in the new SMU architecture provide the same decoding accuracy as a standard Viterbi decoder at 100MHz and the timing skew, as discussed before, yields no accuracy degradation. Further testing at 50MHz (where timing skew may cause decoding errors according to the timing analysis) also results in the same BER as the standard decoder at the same frequency. This suggests that the negative and positive timing skew at different stages tend to cancel each other out so that they will not affect the decoding accuracy in the real design.

The power simulations are run for 10k symbols at the $Eb/No = 0dB$ with a 50MHz clock. Only the dynamic power is considered since the quiescent power consumption in CMOS circuitry is very small. The results are listed in Table 5.10. These power

Tab. 5.10: Dynamic power consumption of the Viterbi decoder with the new SMU Implemented on FPGA

	Trace back length	**Power(mW)**
VD with new SMU(k=3)	48	20.3
Xilinx IP(k=3)	48	28.2
VD with new SMU(k=7)	48	428.3
Xilinx IP(k=7)	48	446.2

figures indicate that for small constraint length, such as $k = 3$, the proposed new SMU saves 28% dynamic power of a standard Viterbi decoder on the FPGA. However, for $k = 7$, the saving becomes only 4%. This is due to the exponential increase in the design complexity mainly from the increased number of trellis connections. For a small constraint length, the increased power dissipation of these extra connections is relatively small. However, as the number of the connections increases exponentially with the constraint length, the power overhead becomes significant and reduces the power saving of this new SMU. This effect is more obvious in FPGA implementations since the connections in FPGA are not simply wires as in CMOS implementations.

For the k=7, L=48 design, further tests of a standard one-pointer synchronous SMU show that by using the low power block RAM on the FPGA, the SMU power dissipation can be reduced from $281mW$ to $274.8mW$. This compares with $263.1mW$ in the new SMU power consumption. So the new design saves only 4% SMU power dissipation. Therefore, this proposed SMU architecture is best suited for low power CMOS implementations of the Viterbi decoder with a relatively small constraint length.

5.5 Summary

In this chapter, a new trace back SMU design has been proposed. Based on the timing analysis of synchronous and asynchronous SMU designs, the timing features of the trace back algorithm have been identified. Due to the convergence of trace backs, the timing control can be simplified as long as the converged path can be maintained by trace backs.

To implement these features, in this new design data is entered into the SMU synchronously while the tracing back to reconstruct the transmitted data is decoupled from the data entry and free runs asynchronously. The trace back logic is entirely combinational thus avoids the complex control and handshakes normally associated with asynchronous design. In addition, the use of latches rather than RAMs for the

data entry avoiding power costly read accesses to the memory. All these measures give the new design an equivalent bit error performance but at lower power than other SMUs.

Potential problems, such as metastability glitch and timing skew, have been shown to not significantly affect the accuracy of the decoding data compared with other more conventional approaches. Although the trace back may fail due to the timing skew, it can be fixed by either shifting the output clock edge or by altering the number of trace back stages. This is confirmed by the measured BERs from the FPGA implementations. This indicates that to achieve an efficient SMU design without handshakes is not without problems that careful timing analysis and simulation is required at the working frequencies and on the targeted process to verify correct operation.

On CMOS circuitry, the new SMU architecture is more power efficient compared with a low power design using SPL as it reduces the power dissipated in the SMU by a factor of 6. Thus, this proposed architecture is believed to be suitable for low power trace back SMU implementations. The FPGA power tests with a standard Viterbi decoder and one-pointer SMU design show the new SMU provides the same decoding accuracy as a standard design; however, the power saving could be significantly reduced for larger constraint length due to the wire overhead. This indicates the proposed SMU architecture is best suited for low power CMOS implementations of the Viterbi decoder with a relatively small constraint length.

6. LOW POWER PMU DESIGN

The power analysis in Chapter 3 indicates that the conventional PMU design consumes 37% Viterbi decoder power. In order to achieve optimum power efficiency in a Viterbi decoder, the PMU power consumption needs to be minimised.

6.1 Existing low power implementations of PMU

Some previous work has been done specifically targeted at lowering power dissipation in the PMU. These works involve using non-discrete approach, e.g. analogue circuits or modifying the ACS process. An analog design approach will be discussed first in this section.

6.1.1 Analog design

Because of their high speed and low power nature, analogue CMOS and BiCMOS integrated circuit technologies are becoming more and more popular [55] [56] in designing many of today's complex, high performance systems. Analogue circuits have already been implemented in some Viterbi decoder designs, mostly in PMU designs [57] [58] [59].

In [60] a mixed signal Viterbi decoder was proposed applying analogue circuits in the BMU and PMU while implementing digital circuits in the SMU. In the BMU, the voltage differences of the received and the ideal symbols are sampled on capacitors in the first clock cycle; and then, in the second clock cycle the charge sharing across these capacitors generates the sum of the voltage differences and gives the voltage outputs as the Branch Metrics (BM). In the PMU, the block partitions of the analogue ACS

processor is exactly the same as the digital ACS processor and every add-compare-select operation is completed in one clock period as well. At first, the Path Metrics (PM) are stored in the form of charges on the capacitors. Then, after the BM is added into the PMs a comparator gives the decision bit while a selector chooses the biggest metric which is then stored in the second clock cycle. To avoid overflow, the PMs are normalised at each cycle by monitoring the largest of all metrics.

The analogue BM and PM in the mixed signal Viterbi decoder design means there is an infinite number of quantization levels. Since there is no information loss during quantization compared with the 3-bit quantization process in a normal digital Viterbi decoder, ideally, an analogue Viterbi decoder should have the highest BER performance due to its infinite resolution.

The ideal analogue Viterbi decoder, as indicated by the simulations from [61], does outperform the digital Viterbi decoders of hard-decision and 2-bit or 3-bit soft-decision. However, its performance is very close to the 4-bit soft-decision digital Viterbi decoder suggesting the ideal analogue Viterbi decoder will not provide significantly better performance than a digital Viterbi decoder using soft-decision of 4-bits or more.

Furthermore, the BER performance of the ideal analogue Viterbi decoder, in fact, cannot be achieved in real hardware practice simply because of the noise and signal distortion in the analogue circuits. In [61], these effects are summarised in three categories:

1. The nonlinear distortion of the branch metric transfer function.

2. The noise introduced by the analogue circuits.

3. The offset of the analogue comparator.

The simulation results from [61] suggest that under certain nonlinear distortion, noise condition, or comparator offset, the analogue Viterbi decoder may provide better BER performance than a 3-bit soft-decision Viterbi decoder. However, since these conditions

may vary in a real hardware implementation, the analogue Viterbi decoder cannot guarantee a better performance in comparison with a 3-bit soft-decision digital design.

The major feature of an analogue decoder is its low power dissipation. In [60], 50mW power dissipation is recorded for the k=7 analogue Viterbi decoder at 20MHz clock frequency. In comparison with the SPL design in [27] of 92.5mW normalised power consumption at the same decoding rate, more than 45.9% power can be saved by the analogue circuits. Since the analogue decoder in [60] is implemented with $0.5\mu m$ CMOS technology while the SPL is in $0.35\mu m$, more power saving can be expected when the analogue decoder is implemented in transistors of a smaller size. Therefore, the analogue Viterbi decoder is obviously more power efficient than a normal digital implementation.

However, there are major issues limiting the practice of implementing analogue circuits in a real design [62].

1. Firstly, an analogue implementation has to be customised and it is difficult to use CAD tools to synthesis or automatically route. Therefore, using analogue circuits in the Viterbi decoder design will cost much more time and design effort than only using digital circuits.

2. Secondly, each analogue design requires a precise model of implementation in order to consider the effects such as the noise or the nonlinearity etc., while the digital design only needs simple timing models for simulation and analysis. Therefore, things are more likely to go wrong when the analogue design is implemented due to the inaccurate modelling.

3. Finally, the dynamic range of the analogue implementation is limited by power supplies and noise. Therefore, the analogue design is less flexible in terms of the working condition and it is impossible to guarantee the performance.

To summarise, an implementation of an analogue Viterbi decoder requires much more design effort and time. Although the analogue decoder can achieve a better power

efficiency, it is less flexible in terms of the working condition and its performance is more uncertain in comparison with the digital design. Therefore, few commercial designs are implemented in analogue circuitry and this thesis will only consider improving the PMU power efficiency with digital circuits.

6.1.2 Low power compare-select-add (CSA) design

In the PMU of a Viterbi decoder, the ACS process requires 4 adders and 2 comparators in the butterfly ACS unit in the conventional architecture. This is shown in Figure 6.1. This architecture can be modified by re-arranging the ACS operations to use less adders.

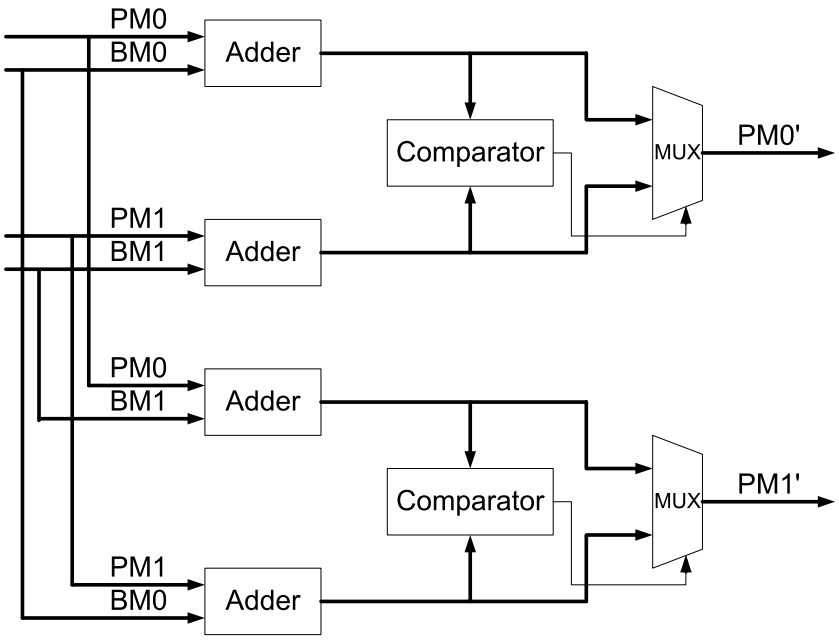


Fig. 6.1: The conventional architecture of the ACS unit.

In [2], a new architecture of the ACS unit is proposed using 2 adders, 2 comparators and 1 subtractor, as shown in Figure 6.2. In this design, instead of adding the PMs with the BMs to produce four candidate PM values, the differences of PMs and BMs are pre-computed and compared so that the minimum of the four candidate PMs can be decided without actually obtaining the values. This architecture from [2] is illustrated in Figure 6.2. Then, based on the decision, the proper PMs and BMs are selected and

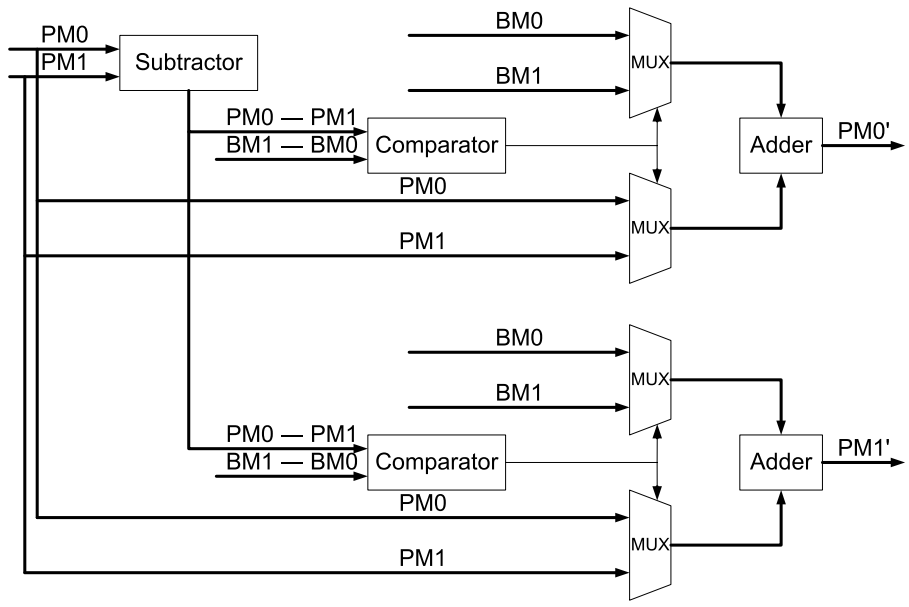


Fig. 6.2: The low power ACS unit architecture proposed in [2].

added together to produce the new PM values. In general, this compare-select-add (CSA) approach avoids the redundant add operations so that two adders can be saved and the power dissipation can be reduced. In [2], 30% power saving is recorded from the gate level simulation. Since the ACS units consume most of the power in the PMU, one third of the PMU power saving can be expected.

The drawback of this CSA approach is the increased delay on the critical path. As indicated in Figure 6.1, the critical path of ACS process includes an adder, a comparator and a multiplexer. However the CSA process in Figure 6.2 involves a subtractor, a comparator, a multiplexer and an adder. The extra subtractor increases the maximum delay of the critical path, and thus reduces the maximum clock frequency and throughput of the PMU implementation.

6.2 Low power design of the PMU

In the Viterbi decoding process, there is only one most likely path and most of the ACS operations are calculating the PM values of the incorrect paths. Therefore, by properly

setting a limit (cap) of the maximum PM value, the number of switching activities in the ACS process can be minimised so that the power dissipated can be saved.

6.2.1 Performance analysis with BM and PM Capping

In the Viterbi algorithm, PMs are used to differentiate and classify the likelihood of the code words paths. Setting a limit of the maximum PM value will not affect the decoding accuracy when the correct path has a PM lower than this limit. The accuracy only decreases when the PM of the correct path is equal or higher than the maximum limit since the correct path may be disregarded randomly by the ACS operation. Moreover, the overall decoding accuracy of the Viterbi decoder is also dependent on the accuracy of BMs. Therefore, systematic simulations are performed in order to reveal the optimum combination of the BM and PM in terms of performance and efficiency. Figure 6.3 shows the BER performance of the R=1/2, k=7 decoder

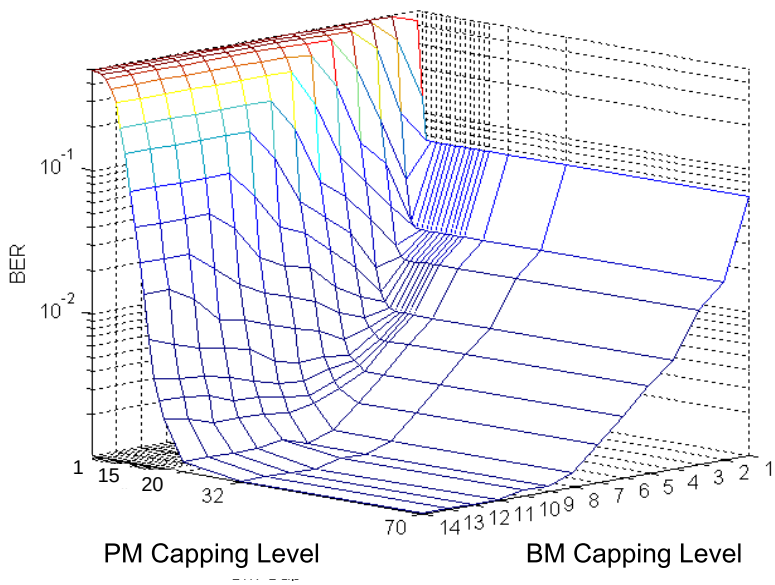


Fig. 6.3: The BER performance variation of the Viterbi decoder with different BM and PM capping levels at Eb/No of 3dB.

with different BM and PM capping levels in a 3D space. It indicates the BER reduces dramatically with the increase of BM and PM capping levels and the minimum BER

can be obtained with the BM and PM at the maximum of 14 and 70 respectively.

Performance analysis with BM capping variation

However, increasing BM capping level does not always result in the performance enhancement. As is shown in Figure 6.4, when the PM capping level is greater than 3, the

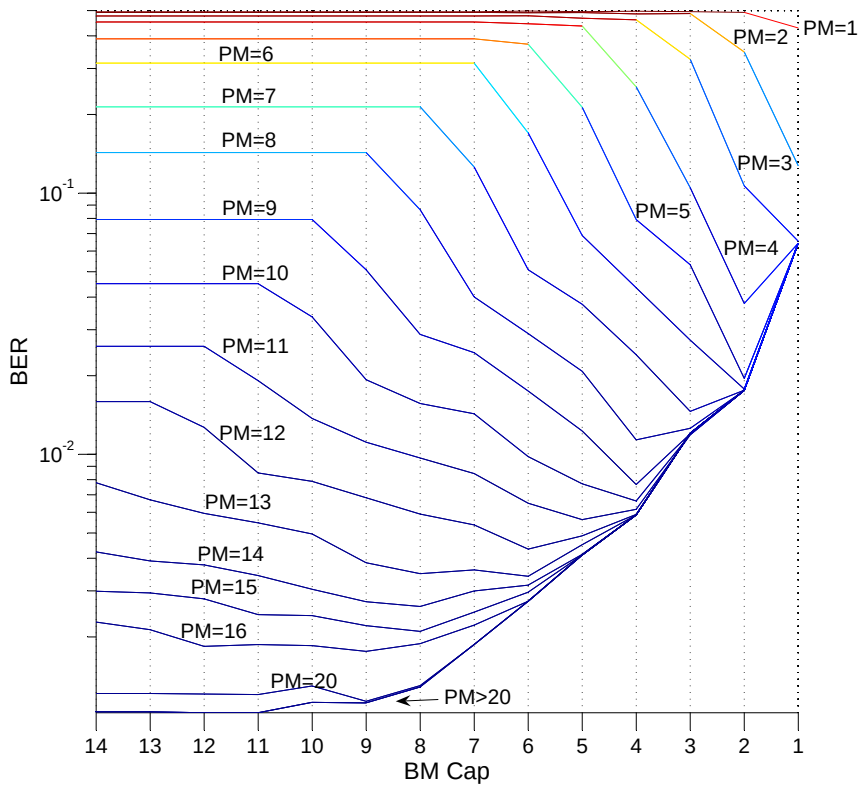


Fig. 6.4: The BER performance of the Viterbi decoder with the variation of BM capping level at Eb/No of 3dB.

BER will decrease first with the increase of the BM capping level; however the BER increases after a certain point as the distance between the BM and PM capping levels reduces. Therefore, it suggests that for each PM capping level there is an optimum BM capping level, with which the optimum performance can be achieved; and when the BM capping level is less than the optimum, the performance of the Viterbi decoder will always improve with an increase of the BM capping level. Therefore, if capping

the PMs in the Viterbi decoder design, the BMs have to be capped at a level equal or less than the optimum BM capping levels; otherwise not only the power is wasted in calculating larger BMs but also the BER performance is decreased.

For R=1/2 k=7 decoder, these optimum BM capping levels at various Eb/No ratios are listed in Table 6.1. This table indicates that the optimum BM caps are determined

Tab. 6.1: The optimum BM caps for R=1/2, k=7 Viterbi decoder.

Max PM Values	Eb/No =3	Eb/No =4	Eb/No =5	Eb/No =6	Eb/No =7	Eb/No =8	Eb/No =9	Eb/No =10
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1
4	2	2	2	2	2	1	1	1
5	2	2	2	2	2	1	1	1
6	3	2	2	2	2	1	1	1
7	3	3	3	2	2	1	1	1
8	4	4	4	2	2	1	1	1
9	4	4	4	2	2	1	1	1
10	4	4	4	2	2	1	1	1
11	5	5	6	2	2	1	1	1
12	6	5	7	2	2	1	1	1
13	6	6	6	2	2	1	1	1
14	8	6	6	2	2	1	1	1
15	8	10	6	2	2	1	1	1
16	9	10	6	2	2	1	1	1
20	9	10	6	2	2	1	1	1
32	11	10	6	2	2	1	1	1
70	11	10	6	2	2	1	1	1

by the PM caps and the noise ratio: generally, a higher PM cap requires a higher optimum BM cap to achieve better performance; however, at higher Eb/No ratios, smaller BM caps are required with greater distances between BM and PM caps. Thus, it can be summarised that the performance of the Viterbi algorithm is not only determined by the absolute values of PMs or BMs but also by the distance between the maximum PMs and BMs. Therefore, the performance of capping BMs and PMs at higher values can also be achieved by capping them at lower values but with larger distances. For example, as shown in Figure 6.4, the BER from capping PMs at 11 and BMs at 11 can

also be achieved by capping PMs at 11 and BMs at 2 or capping PMs at 10 and BMs at 9, or using even smaller values such as capping PMs at 5 and BMs at 2.

Performance analysis with PM capping variation

Although increasing the PM caps will always enhance performance, the decrease in BER varies with different BM caps. This is shown in Figure 6.5 which indicates a major feature of the BER performance variation. For a BM cap, the BER decreases

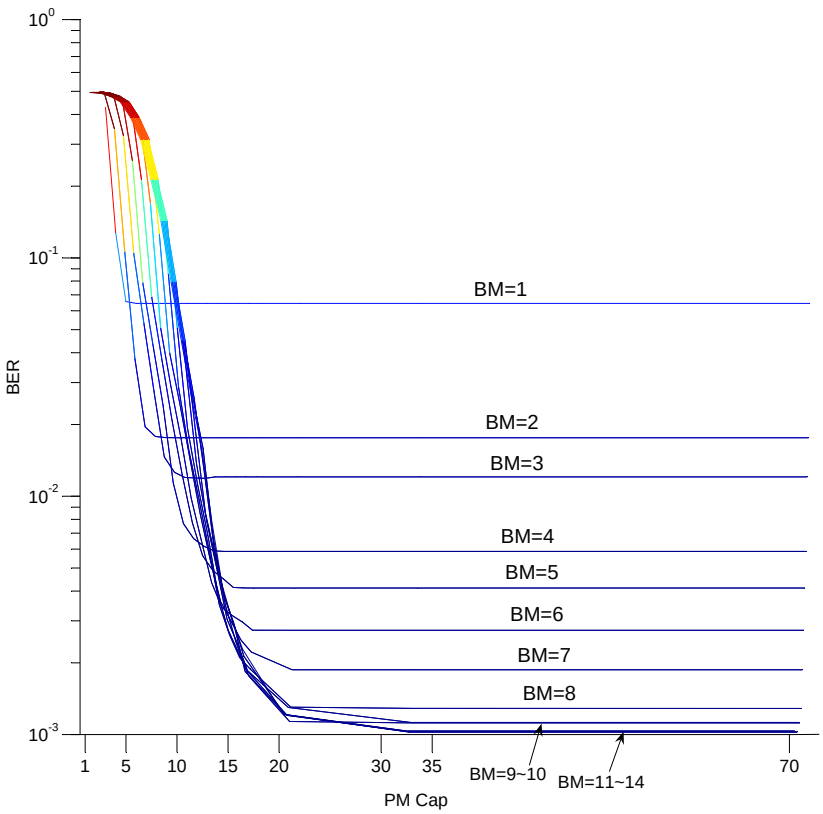


Fig. 6.5: The BER performance of the Viterbi decoder with the variation of PM capping level at Eb/No of 3dB.

with the increase of the PM cap. However, the increase of performance is bounded at a certain PM capping level after which the BER keeps constant. For instance, if capping the BM at 2, the BER decreases rapidly when increasing the PM caps up to

5. After this, the BER remains constant regardless of the increase in the PM caps. This is due to the loss of the likelihood information by the BM capping. In this case increasing the PM beyond 5 is not efficient and thus needs to be avoided in a real design. Table 6.2 lists the optimum PM capping levels for Viterbi decoder R=1/2,

Tab. 6.2: The optimum PM caps for R=1/2, k=7 Viterbi decoder.

Max BM Values	Eb/No =3	Eb/No =4	Eb/No =5	Eb/No =6	Eb/No =7	Eb/No =8	Eb/No =9	Eb/No =10
1	3	1	1	1	1	1	1	1
2	5	1	1	1	1	1	1	1
3	7	1	1	1	1	2	2	
4	9	2	2	2	2	3	3	
5	11	2	2	2	4	4		
6	12	2	2	2	5	5		
7	15	3	3	3	6			
8	16	4	4	7	7			
9-10	20	4	4	8				
11-14	20	4	4	9				

k=7. It can be concluded that in a Viterbi decoder design, if the BM is capped, the efficiency of this design can be achieved by limiting the maximum value of PMs at the values indicated in this table. Therefore, in order to achieve the best efficiency of a Viterbi decoder the optimum BM and PM capping levels should be carefully chosen for the PMU according to the target BER requirement.

6.2.2 Low power implementation of the PMU with PM and BM capping

According to the Digital Video Broadcasting (DVB) standard in [23], the BER of 2×10^{-4} is required from a Viterbi decoder in a DVB system at a Eb/No of 4.5dB. Based on this requirement, the optimum BM and PM capping levels are 6 and 15 respectively according to the previously analysis; this gives a BER performance of 1.78×10^{-4} at a Eb/No of 4dB according to the simulation.

Low power PMU design using the new ACS architecture

Based on the analysis above, in the Viterbi decoder for the DVB application, the maximum BM and PM values should be limited at 6 and 15 so that power dissipation in the BMU and PMU can saved.

With this capping scheme, to minimise the amount of switching in the PMU the adder inputs are frozen by disabling the input register when the input PM values are at the maximum level. This architecture is shown in Figure 6.6 for a R=1/2, k=7 decoder. In this architecture, a BM is limited to 3 bits with a maximum value of 7 while the

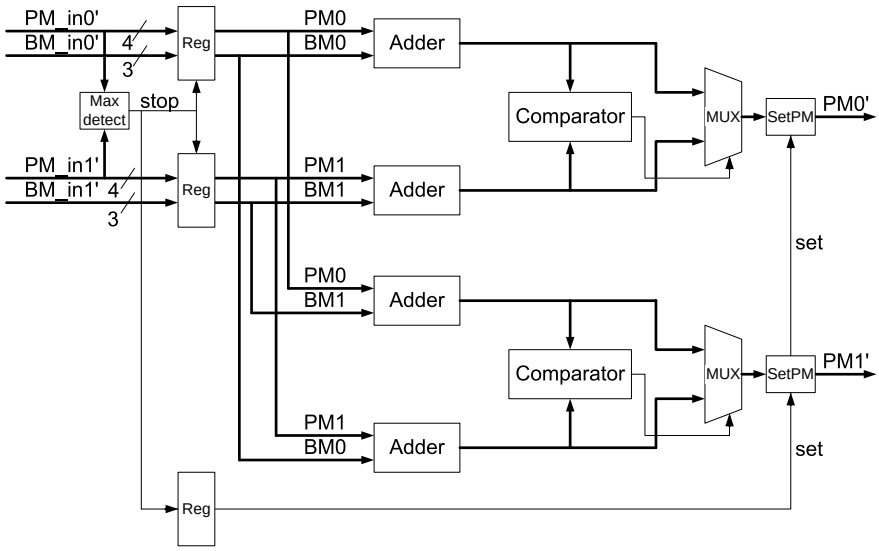


Fig. 6.6: The architecutre of the BM and PM capping ACS unit.

PM is represented with 4 bits and have a maximum value of 15. However when the input PM values are all larger than the PM capping level (15) the Maxdetect logic will disable the registers to freeze the inputs to the 4 adders so that all the ACS operations are avoided for the next clock cycle. Then, in the next clock cycle the PM outputs of the ACS unit are set to the maximum by the SetPM logic without the ACS operation. With this architecture, the ACS power dissipation can be reduced by minimising the number of active bits in the PMs calculations and skipping the ACS process when the PMs are at the maximum. However, the overhead are the extra registers for the BMs

and the logic for monitoring and setting the PM values.

6.3 Test Results

The PMU with this proposed ACS architecture is implemented on a FPGA for R=1/2 k=7 decoder. The design has a maximum frequency of 87MHz and is tested at 50MHz. Figure 6.7 illustrates the BER performance of the Viterbi decoder with the new pro-

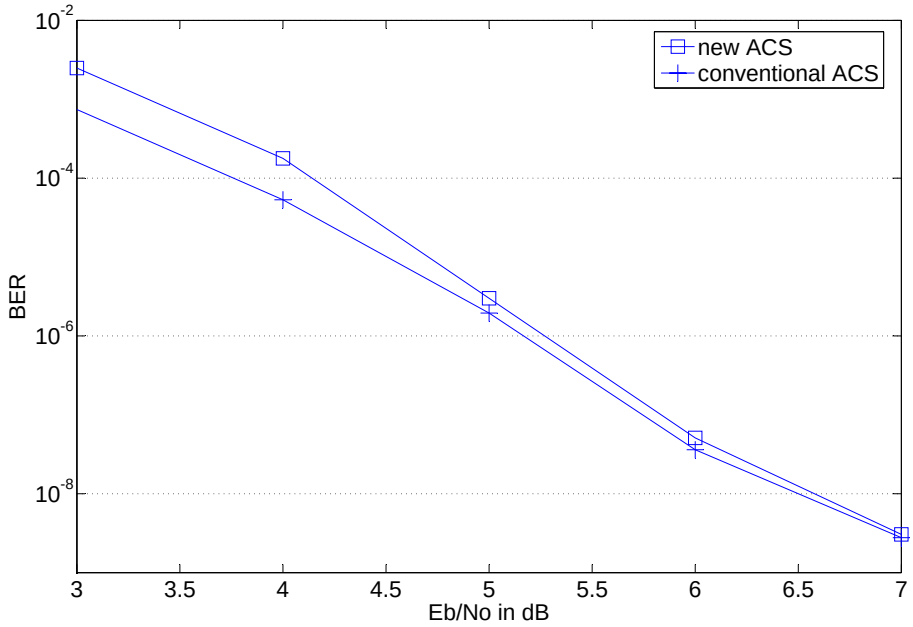


Fig. 6.7: The BER performance of the proposed low power ACS design.

posed ACS architecture in comparison with the normal Viterbi decoder. As is indicated, the new ACS architecture provides lower decoding accuracy (with a $BER = 2.5 \times 10^{-3}$ at 3dB) compared with the standard decoder (with a $BER = 7.8 \times 10^{-4}$ at 3dB), which is approximately 3 times worse. However, as the noise level decreases the difference of the decoding accuracy between the new and the conventional ACS architectures is also reduced. In fact, at the noise level of Eb/No=7dB, as shown in Figure 6.7, the BER figure of the new ACS is very close to the BER of the conventional ACS and the accuracy degradation becomes negligible. This is due to the probability decrease of a correct path with the PM value higher than the capping level. As the noise level goes

down, there are less correct paths being capped at the maximum value thus there are less effects from the PM capping.

The simulated FPGA dynamic power dissipation of a PMU with the new ACS architecture is listed in Table 6.3. The average power reduction of the new architecture is

Tab. 6.3: FPGA simulated power of a R=1/2 k=7 PMU with the new ACS architecture at 50MHz.

Eb/No (dB)	**Conventional ACS (mW)**	**New ACS (mW)**	**Saving (%)**
0	154.6	143.2	7.4%
1	132.9	132.9	n
2	141.7	128.5	9.3%
3	171.7	130.1	24.2%
4	157.6	112.9	28.4%
5	163.7	109.8	32.9%
6	175.3	98.7	43.7%
7	142.8	103.2	27.7%
8	149.4	92.3	38.2%
9	140.4	86.4	38.5%
10	148.9	84.3	42.7%

26.6% compared with a conventinal decoder. Figure 6.8 illustrates the power dissipation in comparison with the conventional PMU. As indicated in Figure 6.8, the power dissipation of the PMU decreases when the noise level reduces. This is because when the noise level decreases the distances between the correct and incorrect paths are increased. Therefore, the chances of the incorrect paths with the maximum PMs are also increased so that more ACS operations can be skipped in the new ACS architecture; thus more power can be saved.

6.4 Summary

In this chapter, the possible options for reducing the power dissipation in a PMU design have been discussed. One of the options is to implement the PMU with analogue circuits. With analogue CMOS circuits, more than 45% power saving can easily be achieved. However, the design is less flexible with analogue circuits in terms of the actual working condition. Moreover, its performance is more uncertain in comparison

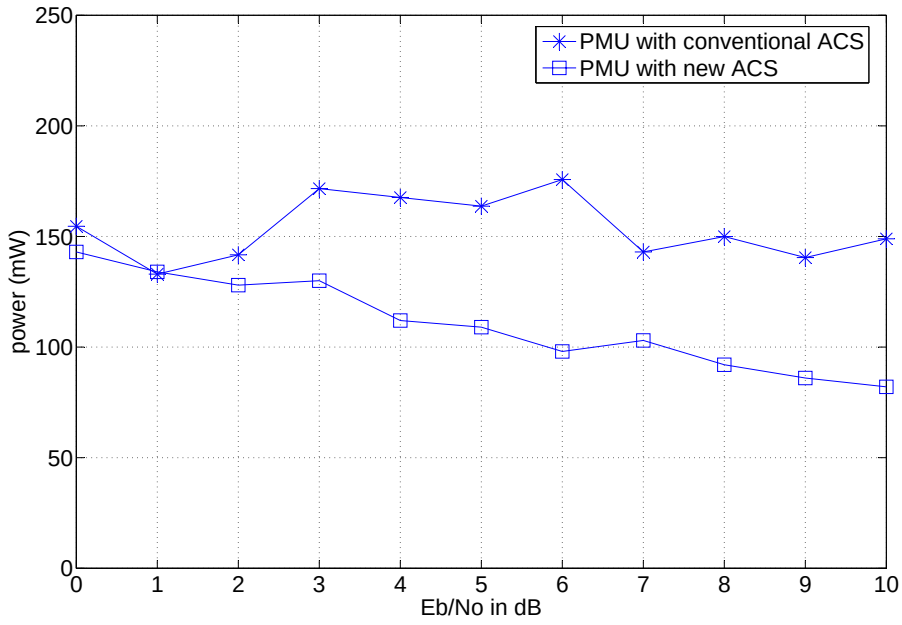


Fig. 6.8: The power dissipation of a PMU with the new ACS architecture.

with the digital design. Therefore, few commercial decoder designs are implemented with analogue circuits and it is not a good approach for improving the PMU power efficiency.

Another option for reducing the PMU power dissipation is to re-arrange the ACS operations. Around 30% power can be save in the PMU in this architecture by reducing the number of adders in the ACS unit. However, the overhead of this scheme is the increased delay on the critical path which reduces the throughput of the PMU implementation.

By analysing the effects over BER performance of the maximum allowed BM and PM values, it reveals that to achieve a particular BER requirement, there is no need to calculate the BM and PM at high values. The path with greater PM is actually less likely to be the correct path. Thus, the ACS operations of a path with a large PM should be avoided. This is achieved by properly capping the maximum values of BM and PM.

Based on this principle, a new ACS unit architecture is proposed for the Viterbi

decoder design for DVB applications. The ACS operations are minimised in two aspects with this new design: the BM and PM are limited at the maximum values of 7 and 15 so that the number of active bits in the ACS operations is reduced; the ACS operations are skipped when the inputs PM are at the maximum. All these measures result in an average power saving of 26% from the PMU with the new ACS architecture. Although this saving comes at the cost of a small loss in decoding accuracy, the performance degradation reduces with the decrease of the noise level. Thus, the proposed ACS architecture is a good technique for reducing PMU power at low noise levels.

In general, the PMU is the most critical part of Viterbi decoder design. It is rather difficult to improve the power efficiency in the PMU since any modification in it will dramatically affect the decoder BER performance. The fundamental method of achieving the PMU power efficiency, as revealed in this chapter, is to carefully select the maximum limits of BMs and PMs based on the BER requirement of the target application. In principle, the efficiency of an IC design can only achieved when it is specifically modified according to the target application.

7. CONCLUSION AND FUTURE WORKS

7.1 Conclusion

This thesis represents the research work of developing new approaches for implementing Viterbi decoder designs to minimize computation complexity and power consumption. This work examines the decoding process of the Viterbi algorithm, the architecture of the Viterbi decoder, and the implementations of the basic functions. This enables the design problems to be discovered. Then a variety of low power design techniques are described and applied to the decoder design to improve its power efficiency.

The new designs are tested by simulations on both software and hardware. The results give a clear view of the improvement of the modifications and enable a novel general methodology for significantly reducing complexity of decoding convolutional codes to be proposed.

7.1.1 Summary of proposed low power Viterbi decoder designs

In a CMOS circuit, power dissipation is dominated by the switching power dissipation. Therefore, the principle of low power designs with CMOS technology is to reduce the number of switching activities in the designs. In order to measure the BER and the power of the Viterbi decoder, the design is implemented on a FPGA device. The dynamic power consumption of the FPGA implementation indicates the switching power dissipation of the design with CMOS technology.

Adaptive approach

The power consumption of the Viterbi decoders with different constraint lengths are measured in the tests. The results indicate that the decoder power consumption reduces gradually with the increase of Eb/No, but exponentially increases with the constraint length. In order to analyse the power efficiency of the decoder with a different constraint length at different Eb/No levels, the power figures are averaged by the number of corrected errors in one second. This gives an energy figure of the decoder in correcting each error. This analysis shows that the energy efficiency of a standard Viterbi decoder reduces dramatically with the increase of the Eb/No and constraint length. Based on this analysis, the fundamental efficiency issue of the Viterbi algorithm can be revealed: the Viterbi algorithm is noise independent so that computational effort could be wasted in processing the error free sequence.

The adaptive approach is one of the major methods for reducing Viterbi decoder power consumption. Existing designs using this approach, however, are basically trading off the decoding accuracy with the power dissipation by approximating and limiting the path metrics and paths number, such as the adaptive T-algorithm. This results in degradation in the decoding accuracy. This is also not efficient since the adaptive capabilities of these approaches are dependent on the method of estimating the error probability of the received sequence and it is difficult to identify the real noise condition and accurate error probability with a simple method.

The ideal approach, which has been revealed in chapter 4, is to transform the Viterbi decoder from channel error independent to error dependent. More precisely, the decoder should be stopped when there is no error in the received code words and restarted to correct errors otherwise. To achieve this, a simple method is required to pre-decode and identify the error free code words sequence. Based on the inverse circuit in [28], a simple approach has been discovered for finding the zero Hamming distance code word path. A new adaptive algorithm, therefore, was proposed so that

the Viterbi decoder was stopped from processing a zero Hamming distance path and the pre-decoded data from the zero Hamming distance path was used as the decoded output instead.

Test results show that with the length of a zero Hamming distance path equal to or larger than 5 times the constraint length, the decoding accuracy of the adaptive algorithm is identical to the standard Viterbi algorithm with the same trace back length. Potential reduction of the power consumption in the Viterbi decoder is from 1.4% to 97% as Eb/No increases from 6dB to 13dB. However, the power reduction is small when the noise level is high ($\leq$6dB).

Novel SMU architecture

In the decoder, the Survivor Memory Unit is the final block. It keeps a history of the computations made on the input data stretching back over many time slots from the current slot. This history enables the SMU to determine and output the most likely data to have been sent out by the encoder at the earliest time slot held by the SMU.

A common approach of decoding the information held by the SMU is to trace back the best path that can be computed based on the information held. In this case, the computations performed may indicate that there are errors in the path(s) being pursued and that the history held requires correction. This gives rise to two concurrent types of activity in the trace back SMU. One is the synchronous inputting of history data which is tied to the external clock and the other is the tracing activity which is best performed asynchronously so that it can free-run as fast as it is able. Furthermore, since tracing through the history time slots is initiated every time new data is received by the SMU, adopting asynchronous timing enables trace back activity with several paths being corrected in parallel if required. This can be perfectly supported by asynchronous timing whereas synchronous timing in the trace back is tied to the clock and usually needs to limit the number of parallel paths which can be traced back at any time. Asynchronous timing for a SMU has previously been implemented [1] but had high

performance and power overhead in the handshake control accompanying the bundled data.

In chapter 5, a new trace back SMU design has been proposed. The timing analysis of synchronous and asynchronous SMU designs revealed that the convergence of the Viterbi algorithm paths through the timeslots provide the opportunity to remove the storage along the handshake path to speed up the operation and lower power. To implement these features, in the new design the local winner data is entered into the SMU synchronously while the tracing back to reconstruct the transmitted data is decoupled from the data entry and free runs asynchronously. The trace back logic is entirely combinational and thus avoids the complex control and handshakes normally associated with asynchronous design. In addition, the use of latches rather than RAMs for the data entry avoids power costly read accesses to the memory. All these measures give the new design an equivalent bit error performance but at lower power than other SMUs. Potential problems, such as metastability glitch and timing skew, have been shown to not significantly affect the accuracy of the decoding data compared with other more conventional approaches.

Although the trace back may fail due to the timing skew, it can be fixed by either shifting the output clock edge or by altering the number of trace back stages. This is confirmed by the measured BERs from the FPGA implementations. This indicates that to achieve an efficient SMU design without handshakes requires careful timing analysis and simulation at the working frequencies and on the targeted process to verify correct operation. On CMOS circuitry, the new SMU architecture is more power efficient compared with a low power design using SPL as it reduces the power dissipated in the SMU by a factor of 6. Thus, this proposed architecture is believed to be suitable for low power trace back SMU implementations. The FPGA power tests with a standard Viterbi decoder and one-pointer SMU design show the new SMU provides the same decoding accuracy as a standard design; however, the power saving could be significantly reduced for a larger constraint length due to the wire overhead. This indicates

the proposed SMU architecture is best suited for low power CMOS implementations of the Viterbi decoder with a constraint length of seven or less.

Low power modifications of PMU

In order to reduce the power dissipation in the PMU, existing designs are either implementing analogue circuitry or re-arranging the ACS operations. With analogue CMOS circuits, more than 45% power saving can easily be achieved. However, the design is less flexible with analogue circuits in terms of the actual working condition. With re-arranging the ACS operations, around 30% power can be saved in the PMU by reducing the number of adders in the ACS unit. However, the overhead is the increased delay on the critical path which reduces the throughput of the PMU implementation.

By analysing the effects over BER performance of the maximum allowed BM and PM values, it reveals that to achieve a particular BER requirement, there is no need to calculate the BM and PM at high values. The path with greater PM is actually less likely to be the correct path. Thus, the ACS operations of a path with a large PM should be avoided. This is achieved by properly capping the maximum values of BM and PM. Based on this principle, a new ACS unit architecture is proposed for the Viterbi decoder design for DVB applications. The ACS operations are minimised in two aspects with this new design: the BM and PM are limited at the maximum values of 7 and 15 so that the number of active bits in the ACS operations is reduced; the ACS operations are skipped when the inputs PM are at the maximum. All these measures result in an average power saving of 26% from the PMU. Although this saving comes at the cost of a small loss in decoding accuracy, the performance degradation reduces with the decrease of the noise level. Thus, the proposed ACS architecture is a good technique for reducing PMU power at low noise levels.

In general, the PMU is the most critical part of Viterbi decoder design. Therefore, it is relatively difficult to improve the power efficiency in the PMU since any modification in it will dramatically affect the decoder BER performance. The fundamental method

of achieving the PMU power efficiency, as revealed in chapter 6, is to carefully select the maximum limits of BMs and PMs based on the BER requirement of the application. In principle, the efficiency of an IC design can only achieved when it is specifically modified according to the desired application.

7.1.2 Contributions and new methods proposed for efficient decoding

In this thesis, the following results can be considered contributions to improve designing and analysing the power efficient Viterbi decoder:

1. **New method of evaluating decoder power efficiency**. To compare the power efficiency of decoder designs is rather difficult since different designs could have different BER performance and throughput. With the measurement of energy-per-corrected-bit used in this thesis, the power efficiency of different designs can be easily compared.

2. **Power efficiency issue of the Viterbi algorithm**. By analyzing the Viterbi decoding process, the error-independency property of the Viterbi algorithm is identified as one of the major problems which affects the power efficiency of the Viterbi decoder design at a high level. More specifically, the Viterbi decoding process is error-independent which means the decoding operation is applied on each one of the received code words without any consideration of their error probabilities. In the situation when a block of received data contains no error, the decoder power is wasted in trying to correct errors in this sequence. Therefore, to improve the power efficiency of a Viterbi decoder, a general methodology is proposed which transforms the decoder from error-independent to error-dependent.

3. **A new adaptive Viterbi decoding approach**. To decode data-dependently means the decoder should run in an adaptive manner. There are some existing adaptive decoding methods for convolutional codes. Most of the adaptability is achieved by approximating the calculation of the likelihood measurement and

correlating the level of decoding operations to the noise level. However, the existing adaptive decoding methods are not efficient due to the difficulty of measuring the noise strength and error probability in practice. Therefore, a new adaptive algorithm is proposed which can detect the sequence which has no error prior to the decoding. Thus the Viterbi decoding operation can be avoided to save power. This adaptive technique has been implemented on a FPGA and demonstrates a significant power saving at low noise levels.

4. **A new SMU architecture**. In a Viterbi decoder, the *Survivor Memory Unit* (SMU) is a vital part of the design. In the conventional trace back implementation, a read-write RAM architecture is generally adopted. However, it suffers from complex control circuits and speed penalty. In this research, a new approach to implement the trace back algorithm targeted at low power applications is proposed. The SMU design based on this new architecture is a mixed synchronous and asynchronous circuit. However, it has no handshake overhead compared to most asynchronous architectures. Post-layout simulation results on a $.18\mu m$ process show the new architecture saves more than 84% of the power dissipated compared with a SMU design using a low power logic family.

5. **Systematic analysis of BM and PM correlation**. In the Viterbi decoder, the PMU consists of different function units. A systematic analysis has been carried out to reveal the correlation between the BM and PM values in terms of their effects on BER performance. Based on the results, modifications, such as capping the branch and path metrics, are proposed to improve the power efficiency in PMU.

6. An AWGN channel test frame has been implemented. In order to simulate the AWGN channel, a noise generator is designed based on the Box-Muller algorithm. This is used to test the standard Viterbi decoders.

7.2 Future works

The review of the information and coding theory in this thesis indicates that encoding the data is a major way of protecting the information in data communication. As more and more research work is carried out in this area, the encoding and decoding process are likely to become more and more complex in the future. The following issues and future research topics have resulted from the research performed for this thesis:

1. According to information theory, to achieve high channel capacity requires more redundancy or more complex computation to maximize code word distances of the codes. However, the theory does not suggest that the decoding process should also be more complex. Therefore, it is necessary to analyse the relationship of channel capacity and decoding complexity. So that the optimum codes can be discovered in terms of both BER performance and decoding complexity.

2. Information theory only quantises information and gives a possible quantity of bits so that information in noise conditions can avoid errors. The theory does not suggest the quantity of computation that is needed to achieve the no-error transmission. Therefore, a theory is needed to give an estimation of the effort used to process a certain amount of information. This could be used to estimate the possible computing complexity of a decoding algorithm. It could also be useful in designing a neuron network to estimate the number of nodes needed for a certain task.

3. Since the maximum likelihood decoding process could be viewed as the classic optimisation problem, it can also be solved using the Hopfield neuron networks. By properly define the energy function of the code words path distance, the most likely path can be identified by the revolving of the neuron network.

4. The ideal of using pre-decoded data from the zero Hamming distance path is decoder independent. Therefore, it could be adopted in other convolutional de-

coding applications, e.g. a Turbo decoder, to minimize power consumption.

BIBLIOGRAPHY

[1] P. Riocreux, L. Brackenbury, M. Cumpstey, and S. Furber, "A Low-Power Self-Timed Viterbi Decoder," in *Proc. Async*, Salt Lake City, Utah, USA, Mar. 2001, pp. 15–24.

[2] C. Y. Tsui, R. S. K. Cheng, and C. Ling, "Low power ACS unit design for the Viterbi decoder," in *Proc. ISCAS '99*, Orlando, FL, USA, May 1999, pp. 137–140.

[3] D. J. C. Mackay, *Information Theory, Inference, and Learning Algorithms.* Cambridge: Cambridge University Press, 2003.

[4] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and Oct. 1948.

[5] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes.* Cambridge: Cambridge University Press, 2003.

[6] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory.* Upper Saddle River, NJ: Prentice Hall, 2002.

[7] G. C. Clark and J. B. Cain, *Error-Corection Coding for Digital Communications.* New York: Plenum Press, 1981.

[8] R. W. Hamming, "Error-detecting and error-correcting codes," *Bell System Technical Journal*, vol. 29(2), pp. 147–160, 1950.

[9] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SLAM Journal of Applied Math.*, vol. 8, pp. 300–304, 1960.

[10] F. J. Macwilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* Amsterdam: ELSEVIER B.V., 1977.

[11] J. H. V. Lindt, *Introduction to Coding Theory.* Berlin and New York: Springer-Verlag, 1982.

[12] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, vol. 4, pp. 37–46, 1955.

[13] (2006) The Official Voyager Website. [Online]. Available: {http://voyager.jpl.nasa.gov/}

[14] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-correcting Coding and Decoding: Turbo-codes," in *Proc. IEEE International Communication Conference*, France, Oct. 1993, pp. 1064–1070.

[15] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. 13, pp. 260–269, Apr. 1967.

[16] J. M. Wozencraft, "Sequential decoding for reliable communications," Research Lab of Electronics, MIT, Tech. Rep. 325, Aug. 1957.

[17] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun.*, vol. 10, pp. 751–772, Oct. 1971.

[18] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. 9, pp. 64–74, Apr. 1963.

[19] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, pp. 675–685, Nov. 1969.

[20] J. L. Massey, *Threshold Decoding.* Massachusetts: M.I.T. Press, 1963.

[21] T. M. Koji Okamoto and A. Yamamoto, "A fully integrated 0.13-μm CMOS mixed-signal SoC for DVD player applications," *IEEE J. Solid-State Circuits*, vol. 38, pp. 1981–1991, Nov. 2003.

[22] (2007) The Official NASA Mission Website. [Online]. Available: {http://www.nasa.gov/missions/highlights}

[23] *Digital video broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services*, ETSI Std. EN 300 421 (v1.12, 1997−08), 1997.

[24] J. M. Wozencraft and B. Reiffen, *Sequential decoding.* Mass.: MIT Press, 1961.

[25] G. C. Cain, R. C. Clark, and J. M. Geist, "Punctured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 97–100, Jan. 1979.

[26] J. L. Massey and M. K. Sain, "Inverses of linear sequential circuits," *IEEE Trans. Comput.*, vol. C-17, pp. 330–337, Apr. 1968.

[27] M. C. Munteanu, "Low power design of integrated circuits," MPhil thesis, The Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, Aug. 2000.

[28] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi Decoder VLSI Implementation and its Performance," *IEEE Trans. Commun.*, vol. 41, pp. 1170–1178, Aug. 1993.

[29] R. Cyper and C. B. Shung, "Generalized trace-back techniques for survivor memory management in the Viterbi algorithm," in *Proc. GLOBECOM*, San Diego, CA, USA, Dec. 1990, pp. 1318–1322.

[30] P. J. Black and T. H.-Y. Meng, "A 140-Mb/s, 32-State, Radix-4 Viterbi Decoder," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1877–1885, Dec. 1992.

[31] H. A. Bustamante, I. Kang, C. Nguyen, and R. E. Peile, "Stanford telecom VLSI design of a convolutional decoder," in *Proc. MILCOM*, Boston, USA, Oct. 1989, pp. 171–178.

[32] I. Kang and A. N. Willson Jr., "Low-Power Viterbi Decoder for CDMA Mobile terminals," *IEEE J. Solid-State Circuits*, vol. 33, pp. 473–482, Mar. 1998.

[33] O. J. Joeressen and H. Meyr, "A 40 Mb/s soft-output Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 30, pp. 812–818, July 1995.

[34] D. Garrett and M. Stan, "Low power architecture of the soft output Viterbi algorithm," in *Proc. IEEE ISLPED*, Monterey, CA, USA, Aug. 1998, pp. 262–267.

[35] P. J. Black and T. H.-Y. Meng, "A 1-Gb/s, four-state, sliding block Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 32, pp. 797–850, June 1997.

[36] Y.-N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-State 10-mW Rate-1/3 Viterbi Decoder," *IEEE J. Solid-State Circuits*, vol. 35, pp. 826–834, June 2000.

[37] N. Weste and K. Eshraghian, *Principles of CMOS VLSI design: a Systems perspective*. Reading: Addison-Wesley, 1993.

[38] G. K. Yeap, *Practical Low Power Digital VLSI Design*. Massachusetts: Kluwer Academic Publishers, 1998.

[39] M. Lewis, "Low power asynchronous digital signal processing," PhD thesis, The School of Computer Science, The University of Manchester, Manchester, Aug. 2000.

[40] A. Chandrakasan and R. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," in *Proc. of the IEEE*, Apr. 1995, pp. 498–523.

[41] Xilinx. (2007, Jan.) Virtex-4 family overview. ds112.pdf. [Online]. Available: http://direct.xilinx.com/bvdocs/publications/

[42] F. Chan and D. Haccoun, "Adaptive Viterbi decoding of convolutional codes over memoryless channels," *IEEE Trans. Commun.*, vol. 45, pp. 1389–1400, Nov. 1997.

[43] R. Henning and C. Chakrabarti, "Low power approach to decoding convolutional codes with adaptive Viterbi algorithm approximations," in *Proc. IEEE/ACM International Symposium on Low power Electronics and Design*, Monterey, CA, Aug. 2002, pp. 68–71.

[44] M. Guo, M. O. Ahmad, M. Swamy, and C. Wang, "An adaptive Viterbi algorithm based on strongly connected trellis decoding," in *Proc. IEEE International Symposium on Circuits and Systems*, Scottsdale, AZ, May 2002, pp. 137–140.

[45] S. Swaminathan, R. Tessier, D. Goeckel, and W. Burleson, "A dynamically reconfigurable adaptive Viterbi decoder," in *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 2002, pp. 227–236.

[46] J. C. Mallinson, *The Foundations of Magnetic Recording.* London: Academic Press, 1993.

[47] C. Lee, "A Viterbi Decoder with Efficient Memory Management," *ETRI Journal*, vol. 26, pp. 21–26, Feb. 2004.

[48] P. J. Black and T. H. Y. Meng, "Hybrid Survivor Path Architectures for Viterbi Decoders," in *Proc. ICASSP*, Minneapolis, MN, USA, Apr. 1993, pp. 433–436.

[49] R. Cypher and C. B. Shung, "Generalized trace back techniques for Survivor Memory Management in the Viterbi algorithm," in *Proc. ICASSP*, Minneapolis, MN, USA, Apr. 1993, pp. 1318–1322.

[50] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective.* Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

[51] S. Ranpara and D. S. Ha, "A Low-Power Viterbi Decoder Design for Wireless Communications Applications," in *Proc. Int. ASIC Conference*, Washington, D.C., USA, Sept. 1999, pp. 377–381.

[52] C. L. Portmann and T. H. Y. Meng, "Metastability in CMOS Library Elements in Reduced Supply and Technology Scaled Applications," *IEEE J. Solid-State Circuits*, vol. 30, pp. 942–951, Jan. 1995.

[53] D. J. Kinniment, A. Bystrov, and A. V. Yakovlev, "Synchronization circuit performance," *IEEE J. Solid-State Circuits*, vol. 37, pp. 202–209, Feb. 2002.

[54] C. Dike and E. Burton, "Miller and noise effects in a synchronizing flip-flop," *IEEE J. Solid-State Circuits*, vol. 34, pp. 849–855, June 1999.

[55] J. C. Daly and D. P. Galipeau, *Analog BiCMOS Design: Practices and Pitfalls.* London: CRC Press LLC, 2000.

[56] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meter, *Analysis And Design of Analog Integrated Circuits.* New York: John Wiley & Sons, Inc., 2001.

[57] T. W. Matthews and R. Spencer, "An integrated analog CMOS Viterbi detector for digital magnetic recording," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1294–1302, Dec. 1993.

[58] M. H. Shakiba, D. A. Johns, and K. W. Martin, "An integrated 200-MHz 3.3-V BiCMOS class-IV partial-response analogue Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 33, pp. 61–75, Jan. 1998.

[59] A. Demosthenous and J. Taylor, "A 100Mb/s, 2.8V CMOS Current-Mode analogue Viterbi decoder," in *Proc. Solid-State Circuits Conference*, San Diego, California, USA, Sept. 2001, pp. 233–236.

[60] K. He and G. Cauwenberghs, "Integrated 64-state parallel analog Viterbi decoder," in *Proc. ISCAS*, Geneva, Switzerland, May 2000, pp. 761–764.

[61] ——, "Performance of analog Viterbi decoding," in *Proc. Midwest Symposium on Circuits and Systems*, Las Cruces, NM, USA, Aug. 1999, pp. 2–5.

[62] P. E. Allen, D. R. Holberg, and Allen, *CMOS analog circuit design.* Oxford: Oxford University Press, 2002.

[63] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," in *Annals Math. and Statistics*, vol. 29, June 1958, pp. 610–611.

[64] D.-U. Lee, W. Luk, J. D. Willasenor, and P. Y. K. Cheung, "A Gaussian noise generator for hardware-based simulations," *IEEE Trans. Comput.*, vol. 53, pp. 1523–1534, Dec. 2004.

APPENDIX

A. MATLAB PROGRAMS

A.1 Matlab program for comparing union bound with simulated BER of Viterbi decoding

```
function [ matrixBER ] = uboundvsSim(EbNo0, EbNo1)
% EbNo points
EbNo = EbNo0:.5:EbNo1;
linEbNo = 10.^(EbNo(:).*0.1);
% QPSK
M=4;
% set code parameters
codeRate = 1/2;
constlen = 7;
codegen = [171 133];
trellis = poly2trellis(constlen, codegen);
% get the uncoded BER
nonCodBER = berawgn(EbNo,'psk',M,'nondiff');
% get the union bound with the first 7 sample
dspec = distspec(trellis, 7);
expVitBER = bercoding(EbNo, 'conv', 'hard', codeRate, dspec);
% % run Monte carlo simulations
maxNumErrs=100;
maxNumBits=10e100;
simVitBer=[];
for i=1:length(EbNo)
    [ber, numBits] = viterbisim(EbNo(i), maxNumErrs, maxNumBits)
    simVitBer=[simVitBer ber];
end
matrixBER=[nonCodBER;expVitBER;simVitBer];
% plot
```

```
semilogy(EbNo, nonCodBER, 'r');
hold;
plot(EbNo, expVitBER, '-+');
% plot(EbNo, simVitBer, '-*');
xlabel('Eb/No (dB)'); ylabel('BER');
title('Union Bound and Simulated BER Performance for R=1/2, K=7
Conv. Code with Hard Decision Viterbi Decoding and QPSK
Modulation');
grid on;
legend('Uncoded','Union Bound', 'Simulated BER', 0);
```

A.2 Matlab program for comparing union bounds of different code rates

```
function [ matrixBER ] = rate_analysis(EbNo0, EbNo1, deci)
EbNo = EbNo0:.5:EbNo1;
linEbNo = 10.^(EbNo(:).*0.1);
M=4;
nonCodBER = berawgn(EbNo,'psk',M,'nondiff');
matrixBER=[nonCodBER];
semilogy(EbNo, nonCodBER, 'r');
hold;
% R=1/3
codeRate=1/3;
constlen=5;
codegen=[37 33 25];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-+');
% R=1/2
codeRate=1/2;
constlen=5;
codegen=[35 23];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-x');
% R=2/3
codeRate=2/3;
constlen=[3 3];
codegen=[7 5 2; 6 1 5];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen(2));
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
```

```
% plot results
plot(EbNo, expVitBER,'-o');
grid on;
xlabel('Eb/No (dB)'); ylabel('BER');
title('Union bounds for R=1/3, 1/2, and 2/3, Conv. Code with
Soft Decision Viterbi decoding and QPSK Modulation');
```

A.3 Matlab program for comparing union bounds of different constraint lengths

```
function [ matrixBER ] = constlen_analysis(EbNo0, EbNo1, deci)
EbNo = EbNo0:.5:EbNo1;
linEbNo = 10.^(EbNo(:).*0.1);
M=4;
nonCodBER = berawgn(EbNo,'psk',M,'nondiff');
matrixBER=[expVitBER];
semilogy(EbNo, nonCodBER, 'r');
hold;
% k=3
codeRate=1/2;
constlen=3;
codegen=[5 7];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-+');
% k=4
codeRate=1/2;
constlen=4;
codegen=[15 17];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-x');
% k=5
codeRate=1/2;
constlen=5;
codegen=[23 35];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
```

```
plot(EbNo, expVitBER,'-*');
% k=6
codeRate=1/2;
constlen=6;
codegen=[53 75];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-s');
% k=7
codeRate=1/2;
constlen=7;
codegen=[133 171];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-d');
% k=8
codeRate=1/2;
constlen=8;
codegen=[247 371];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', deci, codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-^');
xlabel('Eb/No (dB)'); ylabel('BER');
title('Performance for R=1/2, k=3, 4, 5, 6, 7, and 8 Conv.
Code and QPSK with Soft Decision');
```

A.4 Matlab program for comparing union bounds of hard and soft decision

```
function [ matrixBER ] = deci_analysis(EbNo0, EbNo1)
EbNo = EbNo0:.5:EbNo1;
linEbNo = 10.^(EbNo(:).*0.1);
M=4;
nonCodBER = berawgn(EbNo,'psk',M,'nondiff');
matrixBER=[expVitBER];
semilogy(EbNo, nonCodBER, 'r');
hold;
% hard decision
codeRate=1/2;
constlen=7;
codegen=[133 171];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', 'hard', codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-+');
% soft decision
codeRate=1/2;
constlen=7;
codegen=[133 171];
trellis = poly2trellis(constlen, codegen);
dspec = distspec(trellis, constlen);
expVitBER = bercoding(EbNo, 'conv', 'soft', codeRate, dspec);
matrixBER=[matrixBER;expVitBER];
plot(EbNo, expVitBER,'-*');
xlabel('Eb/No (dB)'); ylabel('BER');
title('Performance for R=1/2, k=7 Conv. Code with Hard/Soft
Decision and QPSK Modulation');
```

B. THE AWGN GENERATOR

B.1 The AWGN generator design overview

The AWGN generator is designed based on the Box-Muller algorithm [63], which generates random Gaussian variables by transforming two uniform random variables over (0, 1]. It offers a predictable output rate and, in combination with the central limit theorem, very good Gaussian modelling [64]. According to the Box-Muller method, by given two independent realizations u_1 and u_2 of a uniform random variable over the interval $(0, 1]$ and a set of intermediate function f, g_1, and g_2 such that

$$f(u_1) = \sqrt{-ln(u_1)} \tag{B.1}$$

$$g_1(u_2) = \sqrt{2}sin(2\pi u_2) \tag{B.2}$$

$$g_2(u_2) = \sqrt{2}cos(2\pi u_2) \tag{B.3}$$

the products,

$$x_1 = f(u_1)g_1(u_2) \tag{B.4}$$

$$x_2 = f(u_1)g_2(u_2) \tag{B.5}$$

provide two samples of Gaussian variables over (0, 1) [63].

The top level architecture of the AWGN generator can be partitioned into 4 stages, as shown in Figure B.1.

The first stage involves implementation of the functions f, g_1, and g_2. A direct look-up table implementation for $f(u_1)$ with sufficient resolution for u_1 would require a large number of samples, thus it is not realistic and efficient for our FPGA design. Instead, a method modified from the two-step process in [64], based on nonuniform piecewise linear approximation, is used to implement this function. More details about this implementation are described in the next section.

The second stage involves a sample accumulation process that exploits the central limit theorem to reduce the approximation errors. The accumulator (ACC(2) in Figure B.1) sums two successive random numbers to produce an output every other clock cycle. The central limit theorem calls for a division by $\sqrt{2}$. Since computation of g_1 and

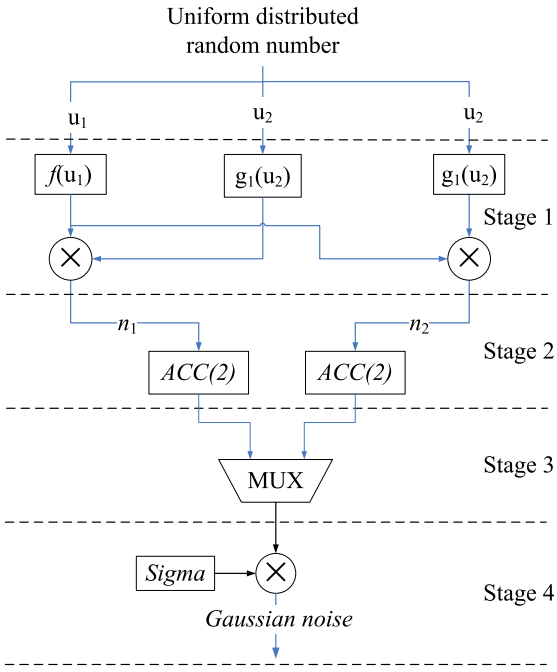


Fig. B.1: Architecture of the AWGN generator

g_2 involve a multiplication by $\sqrt{2}$, this multiplication is cancelled by the subsequent division, so it can be dispensed with in both places in the implementation.

The third stage involves a multiplexer based circuit to select one of the two ACC(2) outputs in alternate clock cycles. Each ACC(2) produces one noise sample in every two clock cycles and the multiplexor selects output between these two ACC(2) every clock cycle. This enables producing an output every clock cycle, rather than two outputs every two clock cycles.

At the last stage, the standard deviation σ, calculated from the specified Eb/No, is multiplied with the noise sample which has the mean of 0 and standard deviation of 1. This produces a random noise sample at the specific Eb/No which has a standard deviation of σ.

B.2 The f function implementation

To effectively implement the f function is the most important part of this design. In a normal approximation approach, the function value is uniformly divided into segments with same x interval, shown in Figure B.2. As Figure B.2 shows, the greatest

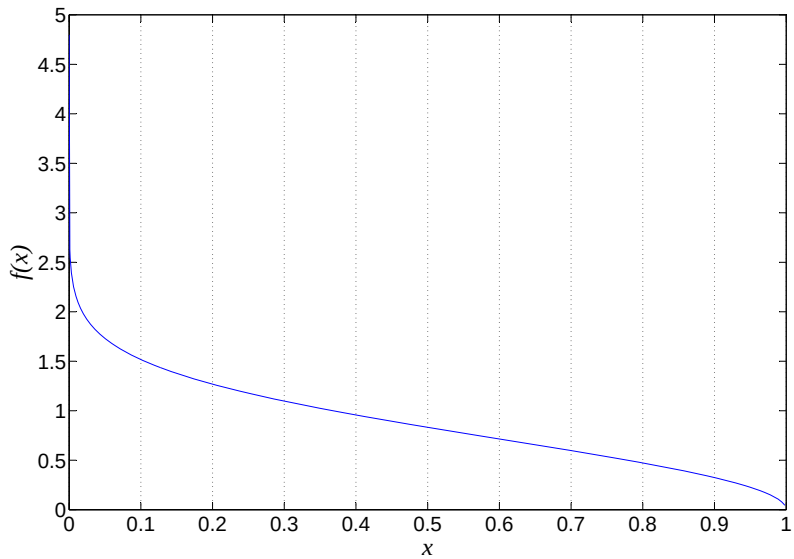


Fig. B.2: Uniform segementation of $f(x)$

nonlinearities of the f function occur in the regions close to zero and one. It is not accurate to approximate segments close to zero and one as linear function with a small number of segments. Therefore, if uniform segments are used, a large number of small segments would be required to get accurate approximations in the nonlinear regions. However, in the middle part of the curve where it is relatively linear, accurate approximation can be obtained using relatively few segments. It will be efficient to use small segments for the nonlinear regions and large segments for the linear regions. A method has been suggested in [64] which can construct piecewise linear approximation such that: 1) the segment lengths used in a given region depend on the local linearity, with more segments deployed for regions of higher nonlinearity, and 2) the boundaries between segments are chosen such that the task of identifying which segment to use for a given input can be rapidly performed.

An example of approximating f with an 8-bit input using this approach can be illustrated in Figure B.3 where the asterisks indicate the boundaries of this approximating approach. This approach uses segments that grow by a factor of 2 from 0 to 0.5 and segments that shrink by a factor of 2 from 0.5 to 1 in the horizontal axis of Figure B.3. In the case of 8-bits input of x, the segment boundaries are at locations

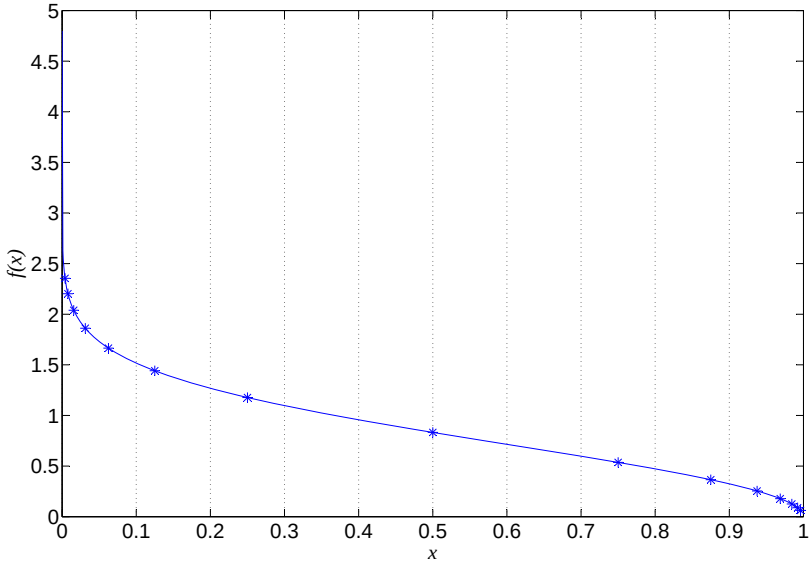


Fig. B.3: A example of 8-bit non-uniform segementation of $f(x)$

2^{n-8} for $0 < x < 0.5$ and $1 - 2^{-n}$ for $0.5 < x < 1$, where $0 \leq n < 8$. The 8-bits binary numbers of the segment boundaries are listed in Table B.1. In our AWGN design, x is a 32-bit random number and the f function is divided into 64 non-uniform segments located by a 6-bit address. A 64 cell ROM is used to store the base value b and segment interval c of $f(x)$. The approximated results are calculated by multiplying the interval c with a 10-bit random number and then adding it to the base b. This architecture is shown in Figure B.4. The result is a two's complement number with 8 bits precision. Therefore, the minimum resolution is 0.00390625 in decimal.

Tab. B.1: Non-uniform segment in 8-bit binary numbers.

Segment NO.	Segments
1	00000000 - 00000001
2	00000010 - 00000011
3	00000100 - 00000111
4	00001000 - 00001111
5	00010000 - 00011111
6	00100000 - 00111111
7	01000000 - 01111111
8	10000000 - 10111111
9	11000000 - 11011111
10	11100000 - 11101111
11	11110000 - 11110111
12	11111000 - 11111011
13	11111100 - 11111101
14	11111110 - 11111111

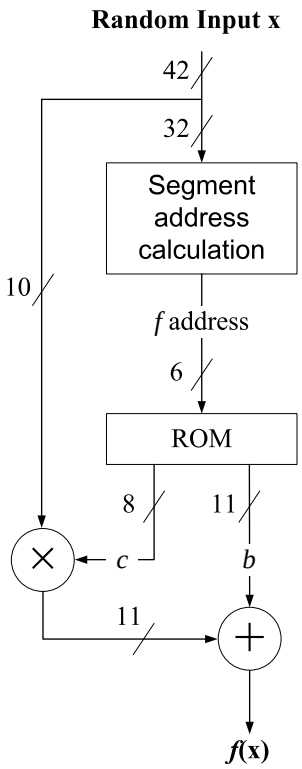


Fig. B.4: $f(x)$ architecture

B.3 The AWGN generator design verification

The BERs of uncoded data are compared with theoretical BER for Eb/No from 0dB to 13dB. The results are closely matched up to 11dB, as shown in Figure B.5. The BER

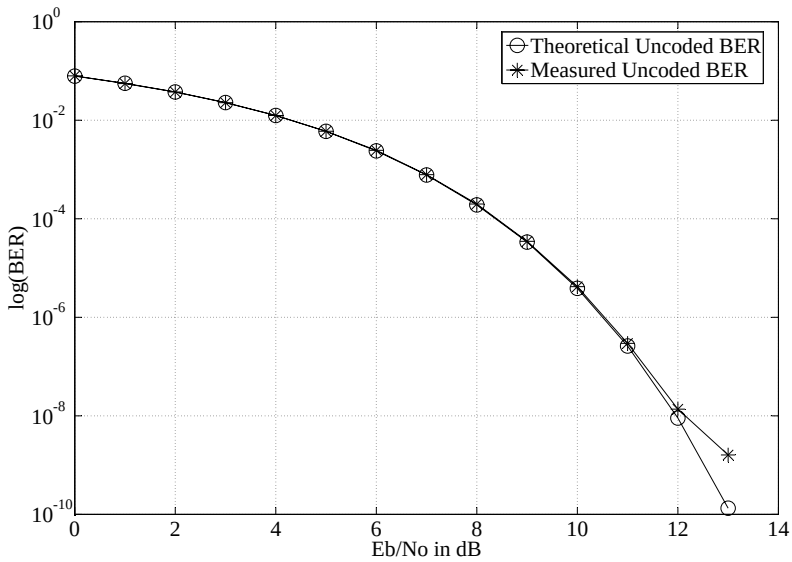


Fig. B.5: Comparison of the uncoded BER

from a standard Viterbi decoder is also tested by this AWGN generator. Figure B.6 shows the BER results from the tests which match the BER specification from the datasheet from 0dB to 5dB. Because of the efficiency of the approximation, the AWGN generator only occupies 11% area of a XC4VLX35 FPGA device, which is 25% smaller than the final design in [64]. To test the Viterbi decoder (1/2 rate) running at 100MHz, two AWGN generators are used in the test frame. Therefore the final design of this test frame is running at 100MHz and occupies 32% of the XC4VLX35 FPGA. Together with the Viterbi decoder, 63% of the total FPGA slices is required. Uncoded data tests with Eb/No from 0dB to 13dB take 45 seconds to finish and the Viterbi decoder tests with Eb/No from 0 to 5 dB take 50 seconds to finish. These are much faster than software simulations.

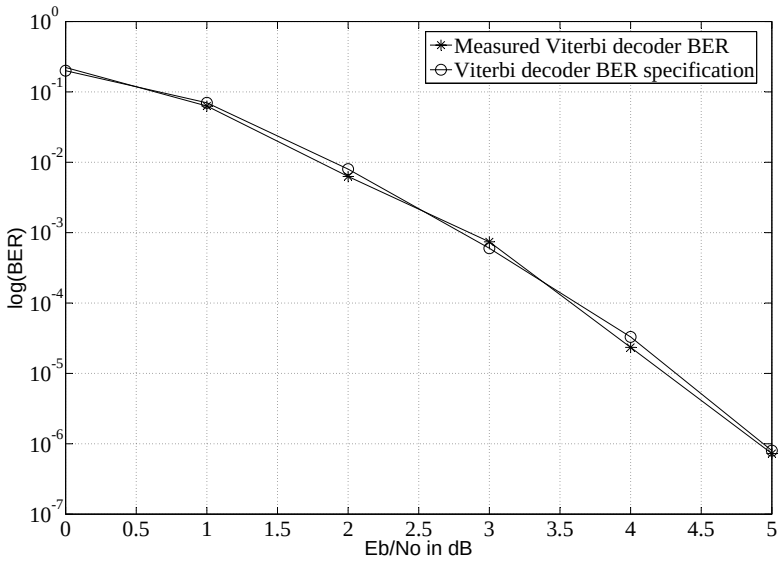


Fig. B.6: Comparison of the decoded BER