Design and Implementation of Low-Power CMOS Analogue Convolutional Decoders Using the Modified Feedback Decoding Algorithm

by

Billy Tomatsopoulos

A thesis submitted for the degree of

Doctor of Philosophy

University College London

April 2007

Department of Electronic & Electrical Engineering

University College London

Torrington Place

London, WC1E 7JE

To my father, mother & uncle

Abstract

Convolutional decoders are very important in digital communication systems, especially in applications where very high noise levels are introduced on the information signal by wireless transmission. Examples of such systems are satellite communications, cellular telephony and digital audio broadcasting (DAB). The most commonly employed decoding method so far in convolutional codes has been the Viterbi algorithm (VA), mainly implemented in digital hardware to accommodate large memory requirements.

In this thesis an introduction to convolutional codes and basic digital communication systems is given, followed by an in depth study of the VA and possible Viterbi decoder (VD) implementations. Advantages and limitations are identified in existing analogue VD designs. A recently proposed algorithm, known as the modified feedback decoding algorithm (MFDA), is then presented and clarified. The MFDA incorporates certain key features of the VA while it requires no digital memory and therefore lends itself to an entirely analogue implementation. This in turn improves performance characteristics, effectively trading complexity and power dissipation against operating speed. The first ever realisation of the novel MFDA is also presented here. Firstly, extensive system-level simulations model errors arising from the use of analogue circuits in practical convolutional decoders. Consequently, and based on these results, a mixed-signal hard-decision modified feedback decoder (MFD) is designed as a proof of principle, using the Austriamicrosystems (AMS) CMOS 0.6μm technology. The fabricated chips have 100% yield and measured results indicate that there is a negligible loss in coding performance compared with a VD. This work can potentially launch the advent of future miniaturised ultra low power analogue convolutional decoders.

Acknowledgments

I would like to thank my supervisor, Dr. Andreas Demosthenous, for his support and guidance throughout my PhD. This work is a direct outcome from the innovative ideas resulting from his PhD thesis. Many thanks to my second supervisor, (and now Prof.) Izzat Darwazeh, for being helpful anytime I needed a second opinion with regards to PhD relevant and irrelevant matters. Also, thanks to Peter Langlois (aka Papous) for his advice and help.

Last but not least, special thanks to everyone who believed in me even when I didn't, and most important thanks to Laura for keeping me alive in the final stages!

Table of Contents

List of Figures and Tables

List of Abbreviations

ACS	Add Compare Select
ACSU	Add Compare Select Unit
AMS	Austriamicrosystems
ARQ	Automatic Retransmission Query
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BCH	Bose Ray-Chaudhuri Hocquenghem
BCJR	Bahl Cocke Jelinek Raviv
BER	Bit Error Rate
BiCMOS	Bipolar Complementary Metal Oxide Semiconductor
BMC	Branch Metric Computer
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
BSVD	Best State Viterbi Decoder
CG	Coding Gain
CMOS	Complementary Metal Oxide Semiconductor
CS	Compare Select
DAB	Digital Audio Broadcasting
DFF	Delay Flip Flop
DMC	Discrete Memoryless Channel
DRAM	Dynamic Random Access Memory
DVB	Digital Video Broadcasting
DVB-S	Digital Video Broadcasting Satellite
DVB-T	Digital Video Broadcasting Terrestrial

ECC	Error Correcting Codes
FD	Feedback Decoder
FDA	Feedback Decoding Algorithm
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
FSVD	Fixed State Viterbi Decoder
GSM	Group Special Mobile
IC	Integrated Circuit
IIR	Infinite Impulse Response
MAP	Maximum a Posteriori
MFD	Modified Feedback Decoder
MFDA	Modified Feedback Decoding Algorithm
MLD	Maximum Likelihood Decoding
NASA	National Aeronautics and Space Administration
NSC	Non Systematic Convolutional
OD	Output Decision
PCB	Printed Circuit Board
PAM	Pulse Amplitude Modulation
PRML	Partial Response Maximum Likelihood
RCC	Replicating Current Comparator
RSC	Recursive Systematic Convolutional
SC	Switched Capacitor
SI	Switched Current
SISO	Serial In Serial Out
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
SP	Sum Product
SS	Symbol Storage
SSM	Survivor Storage Memory
UMTS	Universal Mobile Telecomunication System
VA	Viterbi Algorithm
VD	Viterbi Decoder
VLSI	Very Large Scale Integration
WTA	Winner Take All

CHAPTER 1

Introduction

The increasing demand for reliable and efficient digital communication systems in the last decade has led to a search for new design techniques and strategies in order to accommodate recent trends in technology. This is especially prominent in the area of data transmission and storage that have grown rapidly within the last few years, including applications such as, cellular telephony, digital audio and video broadcasting, satellite communications and hard disk drives. The motivation for research in the field of reliable transmission in communication systems was initiated by Shannon in 1948 [1] when he showed that there is an efficient way to reduce errors encountered in a practical communication channel by means of *encoding/decoding* techniques. Encoding the information sequence prior to transmission means adding extra redundancy to it; this is then used in the receiver end to reconstruct the original sequence, effectively reducing the probability of errors induced by a noisy channel. Different structures of codes have been developed since for what is known as *channel coding*. Convolutional codes [2] are a type of *error correcting codes* (ECC) widely used in channel coding since the late 1960's. They are preferred for their powerful correcting capability and high speed at low cost compared with their competing block codes [3]. Other types include *error detecting codes* which only detect errors and request for retransmission; however these types are more complex and expensive as they require a two way transmission system.

There have been a few convolutional decoding methods [4] from which the most commonly employed technique is the *Viterbi algorithm* (VA) [5]. Dynamic programming (software) and *application specific integrated circuit* (ASIC) design (digital hardware) have been the most predominant ways of realising the VA. However, with the advent of portable devices such as hard disk drives, DVD systems and mobile phones, where size and power dissipation are a prime concern, these methods of realising the VA needed to be revised and reconsidered. *Digital signal processors* or *field programmable gate arrays* (FPGA) can be power hungry and slow whereas dedicated digital ASIC designs provide better performance characteristics. Nevertheless, pure digital implementation can still result in large, complex and power hungry decoders [6]-[7]. Further reduction in size and power dissipation can be achieved using analogue circuit techniques to perform simple computations required in the VA (i.e. addition or multiplication). This was demonstrated in the mid 1990's when BiCMOS and even pure CMOS hybrid digital/analogue Viterbi decoders (VD) begun to appear in the literature [8]-[12]. These decoders were using either current or voltage-mode analogue circuits to realise their computational core improving performance characteristics. The benefits acquired from such implementations were significant resulting in their use in commercial applications such as magnetic recording [8].

Unfortunately, the need for digital memory in the VD imposes a design bottleneck which limits the potential advantages of an entire analogue implementation. This memory is a critical block storing important information related to the nature of the algorithm and can be very bulky, especially as the size of the decoder grows. It has been shown that the memory in an analogue and mixed-signal VD can account for up to 50% of the total die area [11]. Although, the advent of deep sub-micron CMOS technologies nowadays allows for very compact integration of digital circuitry, there are also associated power consumption concerns especially at high speed. On the other hand, in cases where the use of a mature design process and technology is a requirement set by the application, the benefits of entire analogue implementations are tremendous. Such applications include emerging wireless communication systems for implantable biomedical micro-systems, implemented in high voltage processes, where the data rates do not exceed a few MHz [13]-[17].

A more recent convolutional decoding method that requires no digital memory and therefore lends itself to entire analogue realisation was proposed in [18]. This method, called *modified feedback decoding algorithm* (MFDA), can potentially be constructed using simple analogue circuits, trading effectively implementation complexity and power dissipation against operating speed. The MFDA is essentially derived from the *feedback decoding algorithm* (FDA) [19] while it incorporates certain key features of the VA (i.e. path elimination) [20]. Therefore, the resulting algorithm can be thought of as a hybrid FDA/VA combining the merits of both techniques in order to eliminate existing drawbacks.

1.1 Thesis Objectives and Outline

There have been few attempts to replace the digital memory in the VD with analogue counterparts [21], such as *switched capacitor* (SC) circuits [22], none of which have demonstrated any prospective advantage. In fact, besides the need for bulky capacitors, accuracy and linearity issues arise defeating the analogue approach. On the other hand, preliminary system-level simulation results of the MFDA indicate that the latter can achieve a very similar coding performance compared with that of a VA, potentially without the need for any large digital or complex analogue circuitry [18]. This can result in a fully CMOS (i.e. compatible with standard digital CMOS circuits), low-power and low-complexity convolutional decoder suitable for a number of applications of moderate data rates.

The analogue design and implementation of the modified feedback decoder (MFD) using standard CMOS technologies is the main objective of this thesis. In order to achieve the required accuracy, a thorough system-level investigation is necessary to model errors arising from non-ideal analogue circuits and components. Therefore, analysis and simulation for various decoder characteristics is a very important milestone. In the past decade, analogue current-mode techniques have shown to achieve high bandwidth, wide dynamic range and low power operation compatible with modern CMOS technologies [23]. This is especially pronounced in cases where the required processing incorporates simple arithmetic computations (i.e. addition). Consequently, another objective is the search for such circuit candidates and optimum design strategies, also leading to the development of circuits combining basic analogue current-mode and mixed-signal building blocks suitable for a wide variety of low power signal processing applications (e.g. portable devices).

The outline of this thesis is as follows. An introduction covering some basic concepts of *forward error correction* (FEC) with more detail and emphasis on convolutional codes is given in chapter 2. Background information and general applications for three major FEC coding schemes form the basis of this chapter. The VA and MFDA are presented and explained in chapter 3. Similarities and differences are outlined and implementation considerations are also given for both decoders. In addition, system-level simulations using Matlab® (Simulink®) compare their coding performance for various decoder sizes. System specifications and trade offs are extracted for the implementation of various MFDs, accounting for analogue circuit errors and inaccuracies often encountered in practice. Although these simulations are performed assuming *soft-decision* decoding, the results can be interpolated to include *hard-decision* versions by incurring a fixed penalty loss of *coding gain* (CG) of about 2dB. In chapter 4, a brief overview of existing analogue decoders within and beyond convolutional codes is given followed by the design of the first circuit realisation of a hard-decision MFD. Three different decoders are designed in Cadence® using fully CMOS Austriamicrosystems (AMS) 0.8μm and 0.6μm technologies. A mixed-signal design approach is adopted which serves as a proof of the MFDA principle, essentially demonstrating that the same analogue computational circuits with minor modifications can also be used for the construction of an *all*-analogue soft-decision MFD. These modifications and any new circuitry necessary for the implementation of a soft-decision decoder are then described in the end of chapter 4. Finally, conclusions are summarised in chapter 5 where future work is also included. The contents of half the third and the fourth chapters are mainly results stemming from original work.

1.2 Publications

The research work of this thesis has so far resulted in the following conference publications and workshop presentations:

1) Billy Tomatsopoulos and Andreas Demosthenous, "Comparison of the Viterbi and the Modified Feedback Decoding Algorithms in Convolutional Coding", *in Proc. PREP'03,* Exeter, UK, Apr. 2003.

2) Billy Tomatsopoulos and Andreas Demosthenous, "Further Simulation Results of the Modified Feedback Decoding Algorithm for Convolutional Codes", *IEEE Proc. European Conf. Circuit Theory & Design (ECCTD'03),* Krakow, Poland, Vol. 2, pp. 357-360, Sep. 2003.

3) Billy Tomatsopoulos and Andreas Demosthenous, "Design of a Mixed-Signal Convolutional Decoder Based on the Modified Feedback Decoding Algorithm", *Presented in 2nd Analogue Decoding Workshop*, Zurich, Switzerland, Sep. 2003.

4) Billy Tomatsopoulos and Andreas Demosthenous, "Effects of Analogue Implementation Errors in the Modified Feedback Decoding Algorithm", *in Proc. PREP'04*, Hatfield, UK, Apr. 2004.

5) Billy Tomatsopoulos and Andreas Demosthenous, "A Low-Power, Hard-Decision Analogue Convolutional Decoder using The Modified Feedback Decoding Algorithm," *IEEE Proc. Int. Symp. Circuits Syst. (ISCAS)*, Vancouver, Canada, vol. 4, pp. 181-184, May 2004.

6) Billy Tomatsopoulos and Andreas Demosthenous, "Low Power, Low Complexity CMOS Multiple-Input Replicating Current Comparators and WTA/LTA Circuits," *IEEE Proc. European Conf. Circuit Theory & Design (ECCTD'05),* Cork, Ireland, Vol. 3, pp. 241-244, Sep. 2005.

7) Billy Tomatsopoulos and Andreas Demosthenous, "A CMOS Analog Convotutional Decoder Employing the MFDA for Low Power Applications," submitted to IEEE Trans. Circuit. Sys. I, Apr. 2007.

CHAPTER 2

Convolutional Coding and Background

The rapid evolution of digital communication systems in the past two decades emphasises the primary requirement for the use of reliable transmission means and techniques. Existing methods need to be revised and reconsidered while at the same time seeking new more effective ways to achieve appropriate trade offs in terms of cost, complexity and quality. An efficient and economical way of improving the quality of the transmitted information at the receiver end has been the use of *error control coding* [24] since the late 1940's, when Shannon showed that it is possible to predict and model the behaviour of noisy channels. Shannon was the first to introduce statistical models concerning the performance of different transmission sources over communication channels [1] and establish theoretical limits which define the *channel capacity*. These led to the major conclusion of his work which implies that it is possible to reconstruct the transmitted information over a noisy channel, asymptotically error-free at the receiver terminal, by means of appropriate encoding-decoding operation, providing the rate of transmission (defined in symbols per unit time) is below the channel capacity.

There are two major types of error control techniques, the *forward error correction* (FEC) and the *error detection and retransmission*. In the latter category also known as *automatic retransmission query* (ARQ) the decoder only detects errors and requests for retransmission of the sequence [25]. However the use of a two-way link between the receiver and transmitter is not always possible, and in this case the one-way system requirement of FEC coding is very practical. When high reliability is required the combination of FEC and ARQ called the *hybrid* FEC-ARQ system is used exploiting the advantages of the two types while eliminating possible draw-backs of each [26].

This chapter is mainly focused on introductory information about *channel coding* using FEC techniques with more emphasis on *convolutional codes*. The importance of coding in digital communication systems is briefly discussed in section 2.1 where basic concepts and different types of FEC methods are outlined. An introduction to basic *block codes* is given in section 2.2 as a solid background for the detailed description of convolutional codes and applications in section 2.3. Finally, this chapter is closing with a brief outline of the more recent and powerful *Turbo codes* in section 2.4.

2.1 Basic Concepts of FEC Coding

2.1.1 Background Information

In order to understand the purpose of coding it is important to first discuss about the blocks of a simplified digital communication system shown in fig. 2.1. In this diagram the *source* represents the desired information to be transmitted to the *user*. It can be an analogue signal (e.g. audio, video etc.) or digital information (e.g. a digital image or music stored on a cellular phone). In the former case the continuous time waveform needs to be converted to an appropriate digital format compatible with the system and this is done by the *source encoder*, which maps the analogue signal into a digital sequence. In the latter case the use of this encoder can be omitted since the mapping can be direct, however still in most cases its use aims for a more efficient mapping of the source (e.g. data compression). The digital output of the source encoder forms the *information sequence* **u** and in this context it is assumed to be binary. The *noisy*

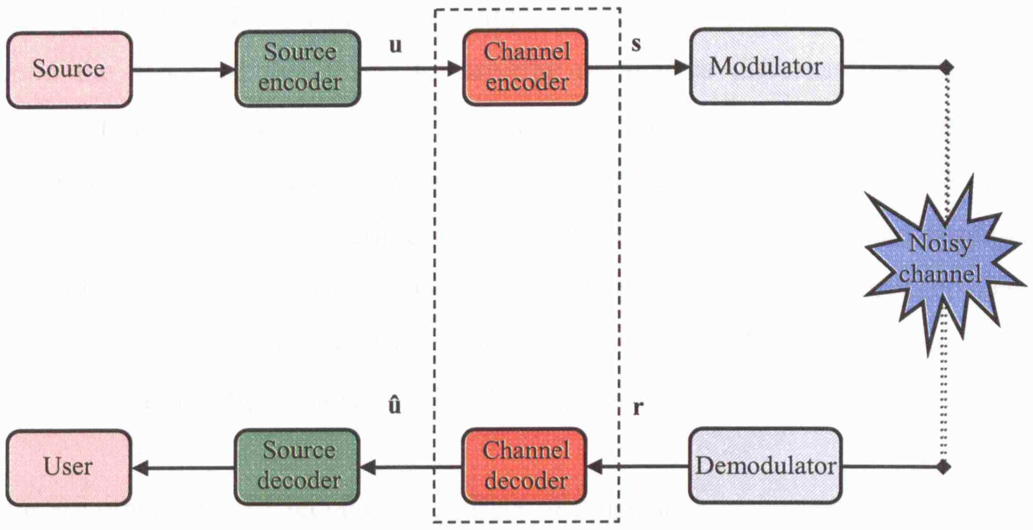


Fig. 2.1 Block diagram of a simplified digital communication system.

channel can be any form of non-ideal communication channel (e.g. coaxial cable fibre optic or the atmosphere, free space etc.) with external disturbance factors (e.g. noise or other interferences) that corrupt the transmitted waveform. There are various sources of noise [3] defined by statistical characteristics using transfer functions to model their behaviour [27]. In order to minimize the effects of noise on the transmitted waveform, the *channel encoder* is used to introduce redundancy (extra bits) by mapping the information sequence **u** into a structured sequence known as *codeword* **s**. The process of converting effectively **u** into **s** in order to minimize the noise effect is known as *channel coding*. Some authors categorise channel coding into two areas, *waveform* coding and *structured sequence* coding. However in this context the term will be restricted to the latter definition. Providing that the structure of mapping, or the *code*, is known in the receiver end, an estimate **û** of the information sequence can be accurately predicted by the *channel decoder* using the codeword. Also, before the transmission, the digital sequence **s** needs to be converted to an appropriate waveform depending on the channel characteristics. This is realised using the *modulator* which can either convert each binary digit into a choice of two continuous-time waveforms representing the two possible states (*binary modulation*), or map k-bit blocks into $M = 2^k$ different waveforms uniquely representing each input block (*M-ary modulation*). There are various modulation schemes that can be adopted depending on the channel characteristics and together with sufficient channel coding, certain design trade offs can be achieved in terms of bandwidth, efficiency, and reliability [28]. In the receiver end a *demodulator* is needed to perform the inverse assignment to the noisy waveform and output the received predicted codeword **r**. It is beyond the scope of this thesis to describe or analyse different modulation and demodulation techniques; more information can be found in [29]. Similarly the *source decoder* is used to decode the estimated sequence $\hat{\mathbf{u}}$ and deliver the desired information to the user. Sometimes modulation and FEC are combined together to form *coded modulation*, which can be very efficient in comparison with channel coding alone as shown in fig. 2.1. Typical examples of such systems include *trellis coded modulation* [30] and *multilevel coded modulation* [31].

There are other ways of reducing the effect of noise such as increasing the power of the transmitted signal or using large antennas. However, the growth of *very large scale integration* (VLSI) techniques for digital signal processing and communications make channel coding and FEC a more beneficial solution at lower costs, exploiting fully the advantages of modern ASIC design trade offs.

2.1.2 Classification of Codes

There are two major types of error correcting codes; these are *block codes* and *tree* or *trellis codes*. Their main difference lies upon the type of encoder used in each case. In block codes, a certain k-symbol input block is mapped into an n-symbol codeword that depends solely on these k symbols. In other words, the encoder of a block code is *memoryless* in terms of its input/output relationship. Tree codes incorporate memory in the encoding procedure where each set of k input symbols produce a set of n symbols, known as *code symbols*, which are determined by the current and a number of preceding input sets. In both cases the rate $R = k/n$, known as the *code rate*, expresses the percentage of added redundancy in a codeword.

Codes can also be classified as *linear* or *nonlinear* depending on the structure and properties of the codewords [4]. Linear codes are used in most practical applications where any codeword is a linear combination of other codewords. This is usually achieved by using modulo-2 adders (or XOR gates) to add two or more symbols in order to produce a code symbol. There are three categories of linear codes, namely block, convolutional and turbo codes. These will be discussed in the following sections.

2.1.3 Hard versus Soft Decision

In order to realise a certain coding scheme a suitable measure of similarity or *distance metric* between two codewords is necessary. This is required by the decoder in order to make the best possible decisions based on comparisons of probabilities representing the likelihood of the received codewords. The two important metrics used to measure the distance between two codewords are the *Hamming distance* and *Euclidean distance* adopted by the decoder, depending on the code scheme, required accuracy, channel characteristics and demodulator type. In practice Hamming distance is used when the transmission is assumed to be over a *binary symmetric channel* (BSC) where each channel symbol is affected independently of the others, and the only distortion that can happen to it is an inversion [5]. In this case, the decoder input is assumed to be quantised into two levels (1-bit) and the decoder is known as *hard-decision*. The former property of the BSC classifies it to a category known as *discrete memoryless channel* (DMC) and its definition can be extended to include the additive white Gaussian noise (AWGN) channel. When transmission takes place over an AWGN channel the preferred metric is the Euclidean distance. The demodulator input is now an analogue waveform and is usually quantised into multiple levels in order to assist the decoder towards a more reliable decision. A 3-bit quantisation resulting in an 8-ary output is common as it will be discussed in chapter 3 and can be visualized in fig. 2.2. In this case, the decoder is called *soft-decision* and as depicted in fig. 2.2 the extra bits (i.e. the two least significant) provide a measure of confidence factor to help identify the received symbol.

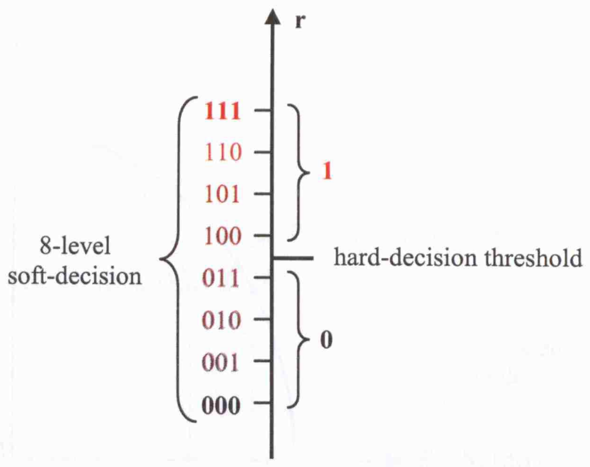


Fig. 2.2 Decoder input for a 3-bit soft-decision versus hard-decision.

2.1.4 Coding Performance Measure

To evaluate the performance of a code a comparative measure of error-rate versus signal-to-noise ratio (SNR) is necessary. This forms the criterion upon decision on "which", "how" and "how much" in terms of code choice, structure and system trade offs, respectively. The common choice for such a quantitative measure is the *coding gain* (CG), which compares the SNR required to achieve a specified bit-error probability P_b with and without coding. In fact a normalised version of the SNR is used, defined as the average received energy per input bit E_b divided by the one-sided noise power spectral density N_o. Also, P_b is defined as the number of bit-errors in a given sequence divided by the total number of bits in that sequence. The CG can be mathematically defined using a theoretical upper bound [32] given by

$$\mathrm{CG} \leq 10\log\left[R\cdot(t+1)\right] \qquad \text{for hard-decision} \qquad (2.1)$$

$$\mathrm{CG} \leq 10\log\left(Rd\right) \qquad \text{for soft-decision} \qquad (2.2)$$

R is the code rate of a t-error correcting code and the terms t, and d are the *error correcting capability* and *minimum distance* of the code respectively and will be explained in the following section. This is visualised in fig. 2.3 where P_b is plotted against E_b/N_o for both uncoded and coded systems. Using this figure and the concept of CG, the designer can control and manipulate certain system specifications or requirements, trading bit-error-rate (BER), transmitted power, data rate and channel capacity versus bandwidth [33]. Finally, practical applications of coded systems show that a soft-decision decoder can achieve a 2 to 3 dB CG improvement over its hard-decision version.

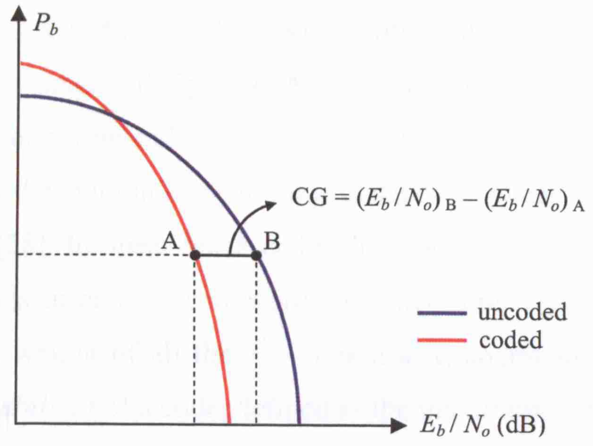


Fig. 2.3 BER curves for typical coded and uncoded systems.

2.2 Linear Block Codes

In block codes the encoder slices the information sequence into blocks of length k and uniquely assigns each k-bit input block into an n-bit codeword introducing redundancy ($n > k$). This assignment can be extended to include symbols from other alphabets in which case k-symbol blocks, where $k \in \{0, 1, \ldots, m-1\}$, are mapped into codewords of length n, where $n \in \{0, 1, \ldots, q-1\}$ and $m \leq q$. However, in this thesis the case of binary linear block codes ($m = q = 2$) will be studied for simplicity and illustration purposes. The resulting 2^k n-bit codewords constitute the so called (n, k) binary block code which is described by the code rate $R = k/n$. As mentioned in section 2.1.2, the encoder of such a code is memoryless, where each codeword is determined exclusively by the current input block. The encoder is usually realised using a set of XOR gates as shown in fig. 2.4, which justifies the linear relationship between the encoder input and output. The mapping illustrated here can also be expressed in a mathematical form as

$$\mathbf{s} = \mathbf{u} \cdot \mathbf{G} \tag{2.3}$$

u and **s** are the information sequence and the codeword row vectors respectively and **G** is the *generator matrix*, which defines the code entirely given by

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & & \cdots & \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix} \tag{2.4}$$

where g_{ij} ($i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, n$) determine the connectivity of the XOR gates and the register stages of fig. 2.4. A very important property of a code that determines its error correcting and detecting capabilities is the *minimum distance* $d_{\min}$, which is defined as the minimum Hamming distance between all possible distinct pairs of codewords [28]. In linear block codes the modulo-2 addition or XOR of any two codewords results in another codeword, and therefore, $d_{\min}$ is equivalent to the minimum Hamming weight of all the $2^k - 1$ non-zero codewords of the code. The *error correcting capability* t of a code, defined as the maximum number of guaranteed single error corrections on each received n-bit codeword, is then expressed as $t = \lfloor (d_{\min} - 1)/2 \rfloor$, where $\lfloor \ \rfloor$ denotes the integer part.

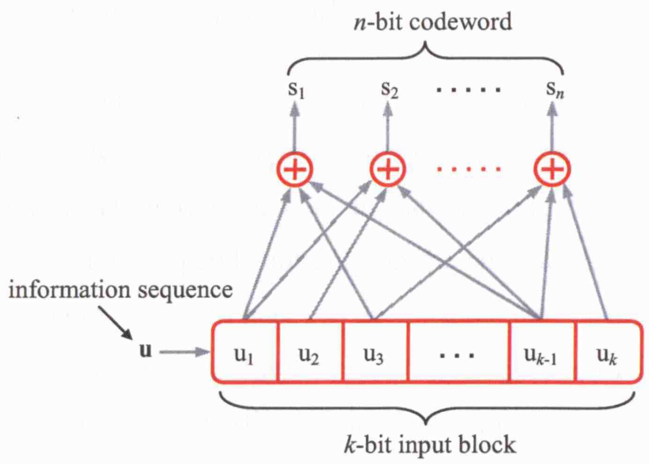


Fig. 2.4 A linear (n, k) block encoder.

Also, an (n, k) block code can detect a number of $e = d_{\min} - 1$ or fewer single errors in a codeword and $2^n - 2^k$ n-bit pattern errors [3].

Decoding of a block code involves the use of the transpose of a *parity check matrix* $\mathbf{H}^T$, which is defined as the $(n - k) \times n$ matrix that fulfils the relation

$$\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}, \tag{2.5}$$

and also determines the sufficient condition for a codeword generated by $\mathbf{G}$:

$$\mathbf{s} \cdot \mathbf{H}^T = \mathbf{0} \tag{2.6}$$

Hard-decision decoding is then performed using a look-up table of all possible received vectors $\mathbf{r}$, known as *standard array*, structured in such a way that the decoder outputs the closest codeword $\mathbf{s}$ to the noisy $\mathbf{r}$. Also, there are several soft-decision decoding algorithms for linear block codes in the literature [34]-[37].

There are many different types of linear block codes from which the most well known are the Hamming codes and the non-binary Reed-Solomon (RS) codes. Hamming codes are characterised by their simplicity and speed and they are widely used in computer networks and memories (DRAM) [38]. They are a special case of binary BCH codes with $d_{\min} = 3$. BCH codes are a powerful family of cyclic codes and their most important non-binary sub-class is the RS codes, which finds numerous applications in space and wireless communications [39], digital magnetic recording and optical storage systems [40]-[41] and cable modems [42].

2.3 Binary Convolutional Codes

Convolutional codes can be seen as a special case of linear block codes in terms of their structure [5]. However they differ in two mutually dependent crucial points: 1) Convolutional codes incorporate memory in the encoding procedure in a sense that each n-symbol code block depends not only on the current but also on previous k-bit input blocks, and 2) the codeword here can be considered to be of infinite length since the information sequence is a continuous real-time sequence of indefinite length. In block codes this sequence is sliced into k-bit input blocks that are encoded independently by a block encoder [4]. In this section, general structures, fundamental properties and commercial applications of convolutional codes will be discussed in detail.

2.3.1 Convolutional Encoding

The general structure of a binary convolutional encoder is shown in fig. 2.5, where k-bit input blocks are inserted serially into a shift register of length $K{\cdot}k$ producing an n-bit code block in each shift. K is known as the *constraint length* of the code and it determines the number of k-bit stages in the encoder or simply its memory size. The convolutional encoder is a *finite-state machine* which features three parameters (n, k, K) and works as follows: every time a new k-bit input block (u_{11}-u_{1k}) is shifted into the encoder, the oldest block (u_{K1}-u_{Kk}) is shifted out, and therefore, each n-bit code block is generated using the current and $K-1$ previous k-bit input blocks. Therefore,

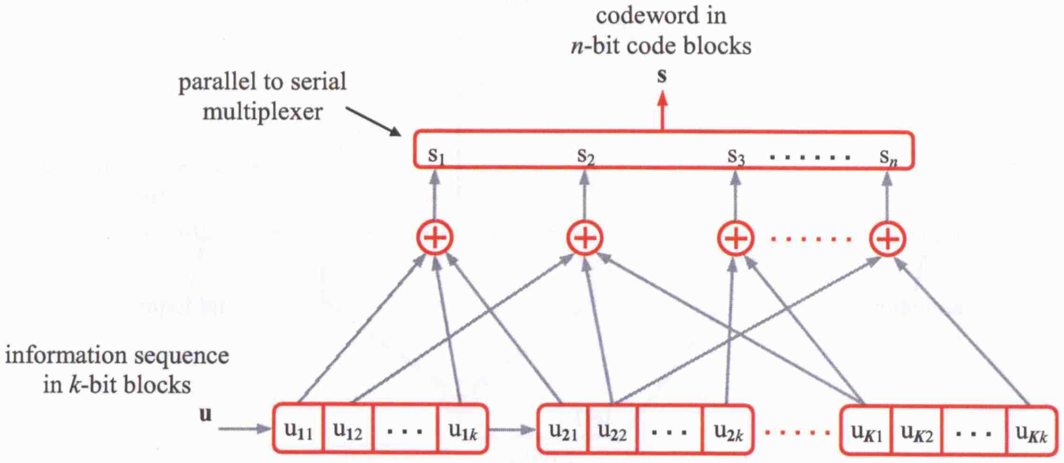


Fig. 2.5 A general binary (n, k, K) convolutional encoder.

the code rate of a convolutional code, defined as the information input bits per code bit, is given by $R = k / n$. The connections to the modulo-2 adders do not follow any specific mathematical pattern though they are not arbitrary since their choice depends on the distance properties of the code that will be discussed in section 2.3.3.

A special case of interest is the convolutional code with $R = 1/n$ where the information sequence is inserted into the encoder bit by bit. A simple (2, 1, 3) encoder commonly used for illustration purposes, which has been proved to be optimum [43] is shown in fig. 2.6. This will be used here to describe a general $(n, 1, K)$ convolutional code in terms of its input **u** and output **s** relationship. The code vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ can be expressed as:

$$\mathbf{s}_1 = \mathbf{u} * \mathbf{g}_1 \tag{2.7}$$

$$\mathbf{s}_2 = \mathbf{u} * \mathbf{g}_2 \tag{2.8}$$

where $\mathbf{g}_1 = [1\ \ 1\ \ 1]$ and $\mathbf{g}_2 = [1\ \ 0\ \ 1]$ are the *generator vectors* following the connections between the modulo-2 adders and the register stages and where the symbol "$*$" means discrete convolution. Also the codeword **s** consists of an infinite number of code bits which are generated by multiplexing the two in this case code vectors $\mathbf{s}_1 = [s_{11}\ s_{12}\ s_{13}\ \ldots]$ and $\mathbf{s}_2 = [s_{21}\ s_{22}\ s_{23}\ \ldots]$ at twice the data input rate, producing $\mathbf{s} = [s_{11}s_{21}\ s_{12}s_{22}\ s_{13}s_{23}\ \ldots]$. Assuming an information input sequence of $\mathbf{u} = [u_1\ u_2\ u_3 \ldots\ u_i\ \ldots]$ (where i is a positive integer) and n generator vectors of the form $\mathbf{g}_p = [g_{p1}\ g_{p2}\ g_{p3}\ \ldots\ g_{pK}]$ (where $p = 1, 2, 3, \ldots, n$), then s_{pi} can be expressed as

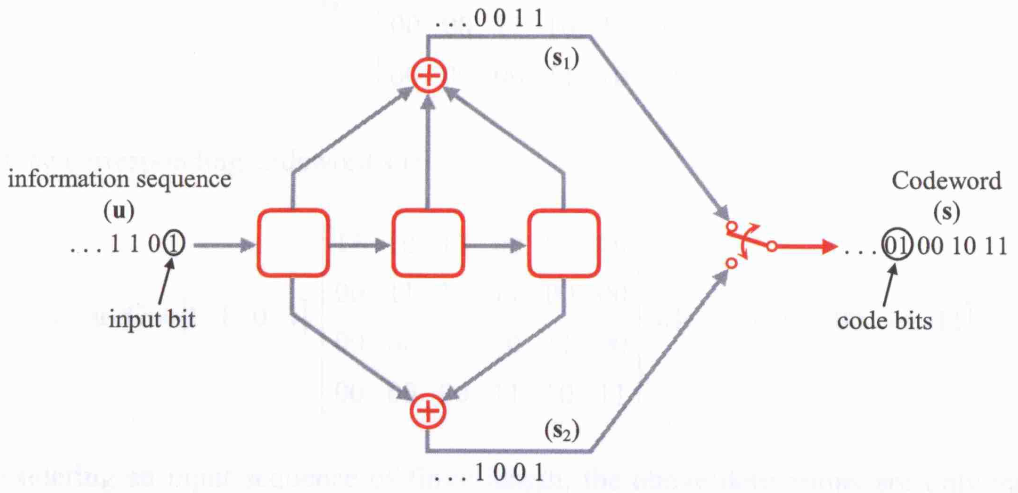


Fig. 2.6 A simple (2, 1, 3) convolutional encoder.

$$s_{pi} = \sum_{x=0}^{K-1} u_{i+x} \cdot g_{p(x+1)} = u_i \cdot g_{p1} \oplus u_{i+1} \cdot g_{p2} \oplus \cdots \oplus u_{i+K-1} \cdot g_{pK} \qquad (2.9)$$

where "·" and "⊕" denote modulo-2 multiplication (logical AND) and addition (logical XOR), respectively. Equation (2.9) depicts an arbitrary code bit from the codeword **s** produced by the p^{th} generator vector and K input bits.

The generator matrix **G** for convolutional codes is determined by vectors $\mathbf{g}_1$ to $\mathbf{g}_n$ and its size depends on the information sequence **u** which is considered at a time. The general structure of **G** has the form

$$\mathbf{G} = \begin{bmatrix} g_{11}g_{21}\cdots g_{n1} & g_{12}g_{22}\cdots g_{n2} & \cdots\cdots\cdots & g_{1k}g_{2k}\cdots g_{nk} & & & \\ & \cdots & & \cdots & & & \\ & & \cdots & & \cdots & & \\ & & g_{11}g_{21}\cdots g_{n1} & g_{12}g_{22}\cdots g_{n2} & \cdots\cdots\cdots & g_{1k}g_{2k}\cdots g_{nk} \end{bmatrix} \qquad (2.10)$$

where all the blank spaces are assumed to be "0"s. Although **u** and **s** are of infinite length, practical realisations of convolutional decoders can not accommodate such a lengthy codeword; instead they segment the received sequence **r** into *decoding windows* of finite length (L) and carry out decoding each block, as it will be described in chapter 3. The generator matrix for the encoder of fig. 2.6 and the information sequence $\mathbf{u} = [1\ 1\ 0\ 1]$ inserted from the left to the right is then

$$\mathbf{G} = \begin{bmatrix} 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 \end{bmatrix}$$

and the corresponding codeword **s** is

$$\mathbf{s} = \mathbf{u} \cdot \mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 \end{bmatrix} = \begin{bmatrix} 11 & 01 & 01 & 00 & 10 & 11 \end{bmatrix}$$

Considering an input sequence of finite length, the above derivations are only valid assuming that the initial condition of the shift register is (0 0 0) and that the information sequence **u** is followed by $K - 1 = 2$ "0"s to clear the contents of the register. Note that including these two "0"s in **u**, the codeword **s** of fig. 2.6 becomes identical to that calculated using **G**.

Another representation of a convolutional encoder is the *polynomial representation* which effectively illustrates the connections of the K register stages to the n modulo-2 adders in terms of n *generator polynomials*. The order of each polynomial does not exceed $K - 1$ and for the encoder of fig. 2.6 they are

$$\mathbf{g}_1(X) = 1 + X + X^2 \tag{2.11}$$

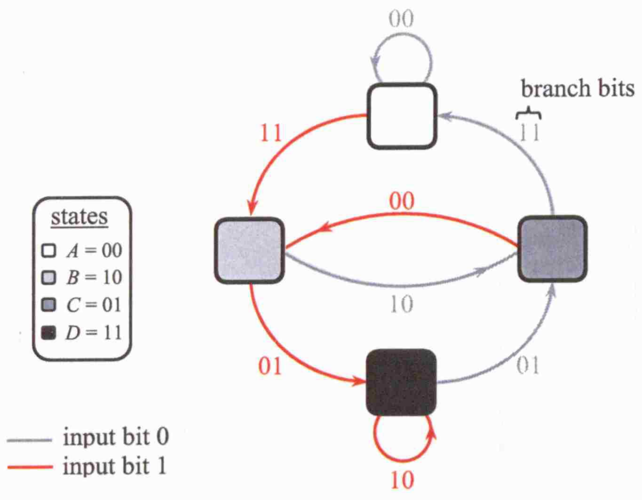
$$\mathbf{g}_2(X) = 1 + X^2 \tag{2.12}$$

The coefficients of these polynomials can either be "1" or "0" depending on whether there is a connection from the corresponding stage to the adder or not respectively. The input sequence can be expressed by an equivalent polynomial representation from which the *code polynomials* $\mathbf{s}_1(X)$ and $\mathbf{s}_2(X)$ can be derived and finally after multiplexing the codeword $\mathbf{s}(X)$ [3].

2.3.2 Representation of Convolutional Codes

In section 2.3.1 basic mathematical formulae describing a convolutional code were discussed using a simple $R = ½$ $K = 3$ encoder. There are other graphical representations that provide more information about a convolutional code. These are the *state transition diagram*, the *tree diagram* and the *trellis diagram*, the choice between which depends on the amount of information and compactness needed [2].

A general (n, k, K) convolutional encoder is essentially a finite state machine (sequential logic circuit) with $2^{k(K-1)}$ different states defined by the contents of the $K - 1$ leftmost registers. Therefore, a state transition diagram would be an obvious way of representing a convolutional code. Fig. 2.7 shows the state diagram for the encoder of fig. 2.6. There are 4 states in this case $A = 00$, $B = 10$, $C = 01$ and $D = 11$, each having $2^k = 2$ different branches entering and 2 emanating to and from the state respectively. Each branch corresponds to a state transition determined by the input bit (u_i) as depicted in the present/next state table of fig. 2.7 and is labelled with $n = 2$ code bits, known as *branch bits* (or *branch word*), resulting from that transition.



Present state	Next state u_i=0	Next state u_i=1	Code bits u_i=0	Code bits u_i=1
00	00	10	00	11
01	00	10	11	00
10	01	11	10	01
11	01	11	01	10

Fig. 2.7 State transition diagram for the (2, 1, 3) convolutional encoder of fig. 2.6.

Another more realistic representation in terms of time history of the input sequence of the convolutional encoder is the tree diagram illustrated in fig. 2.8. The tree shows all possible paths that an information sequence can follow as well as the corresponding states and branch words. The complexity of the tree grows exponentially with time and has 2^t paths, at a certain time instant t, each consisting of t branches. The heavy line indicates the path that the input sequence $\mathbf{u}$ = [0 1 1 0 1] would follow on the tree assuming that it enters the encoder from the left to the right (least significant bit first). Also a red line means an input bit of "1" whereas a grey line means an input bit of "0" and again each branch is labelled with the corresponding transition branch word. Therefore, tracing back the heavy line at anytime would result in the information sequence entered in the encoder and the corresponding codeword produced up until that time. It becomes apparent that the tree diagram is impractical when used to represent the complete route followed by any input sequence of larger length.

Observing the tree diagram, it can be seen that after the first K = 3 levels (t_3) the code structure repeats itself and all 2^K = 4 states have been reached for all possible 3-bit input sequences. Any time step onwards would result in two identical and unnecessary repetitions of the previous step's tree structure. The trellis diagram is a more compact *tree like* representation of a convolutional code taking into account this repetition and for the encoder of fig. 2.6 encountered so far it is illustrated in fig. 2.9. In the trellis diagram there are two branches merging into each state after the first K

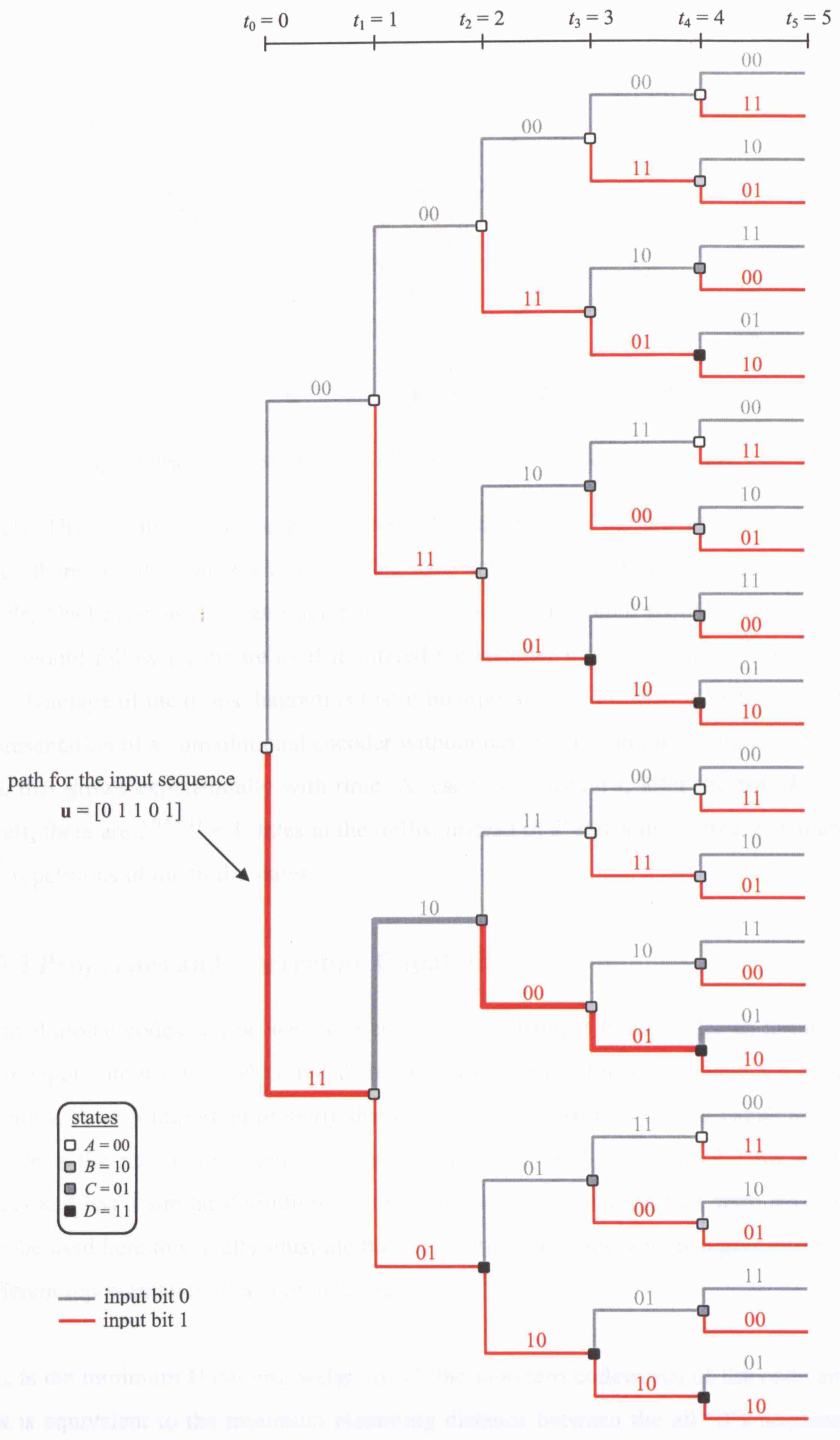


Fig. 2.8 The code tree diagram for the (2, 1, 3) convolutional encoder of fig. 2.6.

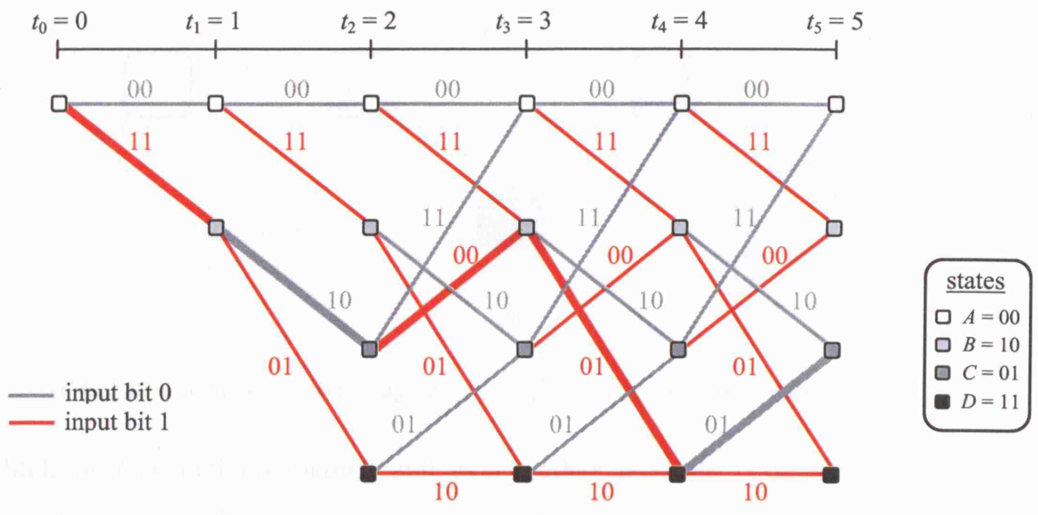


Fig. 2.9 The trellis diagram for the (2, 1, 3) convolutional encoder of fig. 2.6.

levels. These branches emanate from two different states of the preceding time step since there are also two branches leaving each state after the first $K - 1 = 2$ trellis levels. The heavy line indicates again the path that the same input sequence $\mathbf{u} = [0\ 1\ 1\ 0\ 1]$ would follow on the trellis if it entered the encoder with the rightmost bit first. An advantage of the trellis diagram is that it incorporates the dimension of time in the representation of a convolutional encoder without having the complex structure of the tree that grows exponentially with time. At each time instant t, after the first $K - 1$ levels, there are $2^{k(K-1)} = 4$ states in the trellis, instead of 2^t states in the tree which are 2^{t-2} repetitions of the trellis states.

2.3.3 Properties and Correcting Capability

Convolutional codes can be seen as a special case of linear block codes in terms of their input/output relationship and therefore most of their properties have a similar meaning. A very important property that determines the error correcting capability of a code is the minimum distance $d_{\min}$ which was discussed in section 2.2 for block codes and has a similar definition in convolutional codes. A modified state diagram will be used here to visually illustrate the definition of $d_{\min}$ and will also give rise to a different representation of a convolutional code.

$d_{\min}$ is the minimum Hamming weight of all the non-zero codewords of the code and this is equivalent to the minimum Hamming distance between the all "0"s sequence

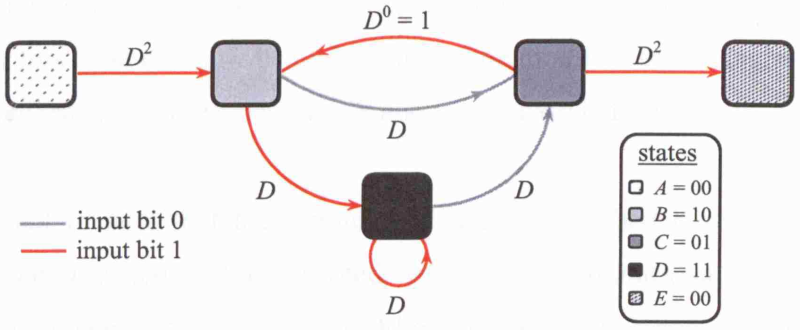


Fig. 2.10 The modified state diagram for the (2, 1, 3) convolutional encoder of fig. 2.6.

(which is a special codeword) and every other possible codeword [2]. Fig. 2.10 illustrates the modified state diagram where the same transition branches as in fig. 2.7 are shown, labelled with the Hamming weight of each branch word expressed in terms of the exponent of D (where D^y denotes a weight of y). The self-loop of state A is eliminated and state A is split into two different nodes A and E in order to exclude the all "0"s codeword and determine $d_{\min}$ by definition. Entering state E illustrates a path that remerges to state A at some arbitrary time instant. Therefore, state A represents the input and state E the output of the modified state diagram. This leads to the definition of the *transfer function* $T(D)$ of the code, which can be easily derived using fig. 2.10. For an infinitely long codeword it is [2]

$$T(D) = \frac{D^5}{1-2D} = D^5 + 2D^6 + 4D^7 + \cdots + 2^j D^{j+5} + \cdots \tag{2.12}$$

Expression (2.12) shows that there is one path with Hamming weight $H_w = 5$ that merges to state A at some time instant. Also, there are two paths with $H_w = 6$ and four paths with $H_w = 7$ and they all merge to state A at different time instants. A more efficient expression that provides extra information about the decoded bits in error and the length of codewords that merge with state A is given by [2]

$$T(D) = \frac{D^5 L^3 I}{1 - DL(1+L)I} = D^5 L^3 I + D^6 L^4 (1+L) I^2 + \cdots + D^{j+5} L^{j+3} (1+L)^j I^{j+1} + \cdots \tag{2.13}$$

The modified transfer function (2.13) of the code indicates that the unique path with $H_w = 5$ merges to state A at t_3 (i.e. it has a length of three indicated by the exponent of L), looking at the trellis diagram of fig. 2.9 and results in one decoded bit in error (indicated by the exponent of I). The two paths with $H_w = 6$ merge to state A at t_4 and

t_5 respectively, while both result in two different decoded bits in error. The minimum distance for this code can now be defined using the modified state diagram as the path with the smallest H_w that reaches state E and in this case is five.

It is important to note that the minimum distance described here is known as the *minimum free distance* and is denoted as d_{free} in convolutional codes where an infinitely long codeword is considered. When the codeword length is truncated into K (as in block codes) the same definition of minimum distance is denoted as d_{min}. Hence, the error correcting capability t of a convolutional code can be expressed as [24]

$$t = \left\lfloor \frac{d_{\text{free}} - 1}{2} \right\rfloor \tag{2.14}$$

where $\lfloor \ \rfloor$ denotes the integer part. Therefore, for the (2, 1, 3) convolutional code example considered so far the error correcting capability is $t = 2$, which means that a maximum of two single input bit errors can be corrected within a certain length L of the codeword. L depends on the error pattern and is usually three to five constraint lengths. Heller showed in [44] that for a $1/n$ short-constraint-length convolutional code, d_{free} is upper bounded by

$$d_{\text{free}} \leq \min_{L \geq 1} \left\lfloor \frac{2^{L-1}}{2^L - 1}(K + L - 1)n \right\rfloor \tag{2.15}$$

More information on free distance bounds for time varying and fixed convolutional codes can be found in [45].

A very important class of convolutional codes is the one in which each k-bit input block is a part of the corresponding n code bits; that is, the n-bit code word produced by a certain k-bit input block, consists of these k input bits and $(n - k)$ parity bits. These codes are known as *systematic* convolutional codes and generally incorporate smaller values of d_{free} than the *non-systematic* codes of the same R and K. It can also be shown [5] that for large constraint lengths, the performance of an $(n, 1, K)$ systematic convolutional code is the same as that of the $(n, 1, K/2)$ non-systematic code, which is obviously less complex to implement. However, the advantageous aspect of systematic codes will be explained after the definition of a disastrous error propagation event which occurs in some convolutional codes known as *catastrophic* convolutional codes. A code is known to exhibit *catastrophic error propagation* when a finite number of received bits in error cause an infinite number of decoded bit errors. The necessary and sufficient condition for catastrophic failure to occur in a $1/n$ convolutional code is when the generator polynomials (described in section 2.3.1) have a common factor [46]. This is equivalent to having a self-loop on the modified state diagram with an all "0"s branch word ($H_w = 0$). An example of a catastrophic code is the (2, 1, 3) convolutional code with generator polynomials $\mathbf{g}_1(X) = 1 + X^2$ and $\mathbf{g}_2(X) = 1 + X$, where there is a common factor of $1 + X$. Systematic convolutional codes never exhibit catastrophic failure since by definition they all have the generator polynomial $\mathbf{g}_p = 1$ (which can not be factorised) or equivalently there is no self-loop in their state diagram with $H_w = 0$. Therefore, non-systematic and non-catastrophic convolutional codes are generally preferred with maximum d_{free} and will be referred to as *optimum* codes in this context. A variety of optimum short constraint length convolutional codes can be found in [47]-[49].

2.3.4 Different Structures and Applications

There are other structures and coding techniques that form excellent channel coding schemes in many practical applications. These structures include *punctured convolutional codes*, which effectively increase the rate R of a code by deleting periodically a code bit [50] and *concatenated codes*, which combine two different code structures (usually an RS outer code with an inner binary convolutional code) to produce a powerful code [51]. Also, another very important method, which was partly developed by Ramsey [52], is the convolutional *interleaving* scheme that finds many applications especially in channels where the errors occur in bursts [53] (i.e. channels with memory). This technique uses an *interleaver* fed straight from the encoder, which is essentially a bank of registers which separates effectively each code bit from another by changing their order in time before they are sent over the channel. A synchronised *deinterleaver* is used in the receiver end, which performs the inverse operation [3] and resembles the original code sequence before fed to the decoder.

Typical applications of general convolutional codes described so far include space, satellite and digital mobile communication systems, digital broadcasting and voice-band data systems. An overview of each application is given in the following sections.

2.3.4.1 Deep space applications

In deep space communications the power of the transmitted signal is the most severe limitation while there is no bandwidth restriction, and therefore, complicated coding schemes can be used to provide large error correcting capabilities. Convolutional codes have been used in many National Aeronautics and Space Administration (NASA) missions since the late 1960's [54]. Table 2.1 summarises characteristics of the codes used in *Pioneer 9-11*, *Voyager* and *Galileo* projects launched for solar orbit, Jupiter and Saturn missions and exploration of other planets [55]. In Voyager and Galileo missions the convolutional codes shown in the table were also combined with an outer RS code in concatenation in order to improve their performance. The RS (255, 223) code was concatenated with the inner (3, 1, 7) convolutional code when Voyager was transmitting images of Pluto and Neptune to earth [56]. The same RS code was used in concatenation with the (4, 1, 15) convolutional code employed by Galileo spacecraft [57].

Table 2.1 NASA missions' convolutional encoding characteristics [54], [57].

Mission	Launch Date	R	K	Generators vectors	d_{free}	Maximum data rate
PIONEER 9	1968	1/2	25	$\mathbf{g}_1 = (40000000000)_8$ $\mathbf{g}_2 = (71547370000)_8$	11	521b/s
PIONEER 9-11	1972-1973	1/2	32	$\mathbf{g}_1 = (35565573735)_8$ $\mathbf{g}_2 = (25565573735)_8$	23	2Kb/s
VOYAGER	1977	1/2	7	$\mathbf{g}_1 = (171)_8$ $\mathbf{g}_2 = (133)_8$	10	100Kb/s
		1/3	7	$\mathbf{g}_1 = (133)_8$ $\mathbf{g}_2 = (171)_8$ $\mathbf{g}_3 = (165)_8$	15	
GALILEO	1989	1/4	15	$\mathbf{g}_1 = (46321)_8$ $\mathbf{g}_2 = (51271)_8$ $\mathbf{g}_3 = (63667)_8$ $\mathbf{g}_4 = (70535)_8$	35	Variable

2.3.4.2 Satellite applications

Applications in satellite communications can be more challenging in terms of high performance and high data rate requirements since both power and bandwidth impose significant limitations. The fixed INTELSAT and the mobile INMARSAT satellite systems [58] employ mainly R = ½ and R = ¾, K = 7 convolutional codes of different data rates within 0.6 and 64Kb/s [59].

2.3.4.3 Digital mobile applications

Convolutional codes are extensively used in digital mobile communication systems such as Group Special Mobile (GSM), *telecommunication industry association interim standard* 136 and *ministry of posts and telecommunications* in cellular telephony of Europe, North America and Japan respectively [60]. In GSM a combination of different block (cyclic) and the (2, 1, 5) convolutional codes together with interleaving techniques and puncturing are typically used, operating at data rates of 2.4 to 9.6Kb/s for data traffic channels and 5.6 to 13Kb/s for speech traffic channels [61].

2.3.4.4 Digital broadcasting applications

Digital broadcasting evolved in the early 1990's improving significantly the transmission of audio and video information signals. Nowadays, both digital audio (DAB) and video (DVB) broadcasting employ convolutional coding in their error correction scheme. Specifically DAB uses a powerful *rate compatible punctured* convolutional code [62] where a higher code rate $R_p > R$ can be achieved from a mother code which is decoded using the same decoder. Also, Terrestrial (DVB-T) and satellite (DVB-S) DVB utilize an inner convolutional code with puncturing concatenated with an outer modified (255, 239) RS code [63]. Table 2.2 summarises the characteristics of convolutional codes employed by DAB, DVB-S and DVB-T.

2.3.4.5 Voice-band data applications

Convolutional codes also find applications in voice-band data communication systems such as modems used in the general switched telephone network. Under bandwidth limited conditions (300-3400Hz) multilevel coded modulation techniques [30]-[31]

Table 2.2 Convolutional encoding characteristics employed in digital broadcasting [63], [64].

Service	R	K	Generators vectors	R_p	Data rate (Mb/s)
DAB	1/4	7	$g_1=(133)_8$ $g_2=(171)_8$ $g_3=(145)_8$ $g_4=(133)_8$	1/2, 1/3	2.304
DVB-T	1/2	7	$g_1=(171)_8$ $g_2=(133)_8$	1/2, 2/3, 3/4, 5/6, 7/8	26-54
DVB-S	1/2	7	$g_1=(171)_8$ $g_2=(133)_8$	1/2, 2/3, 3/4, 5/6, 7/8	Variable

have been used to achieve the required performance without lowering the data rate. During 1984-1994 the *international telephone and telegraph consultative committee*, later called *international telecommunication union telecommunication standardisation sector*, produced three major standards for voice-band data modems namely V.32, V.32 *bis* and V.34 [54]. They all featured trellis coded modulation [30] combined with non-linear convolutional coding schemes [65] at transmission rates of 9.6, 14.4 and 28.8Kb/s respectively. A more recent discovery of other systematic convolutional codes [66], suitable for high-speed voice-band data modems, formed the standard for the V.34 *bis* modem achieving data rates up to 33.6Kb/s [67].

2.4 Turbo Codes

Turbo codes can be considered as a powerful sub-class of convolutional codes. Their encoding procedure consists of a parallel concatenation of some class of systematic convolutional codes called *recursive systematic convolutional* (RSC) codes. Turbo codes have been recently proposed [68] as a new, more complicated coding scheme that can achieve a BER performance approaching the Shannon limit that had never been reached before. Advances in VLSI systems have allowed for their complicated structure and nowadays, just a decade after their proposal, they are extensively used in numerous applications providing an enormous error correcting capability. Turbo codes can best be described following the definition of RSC codes.

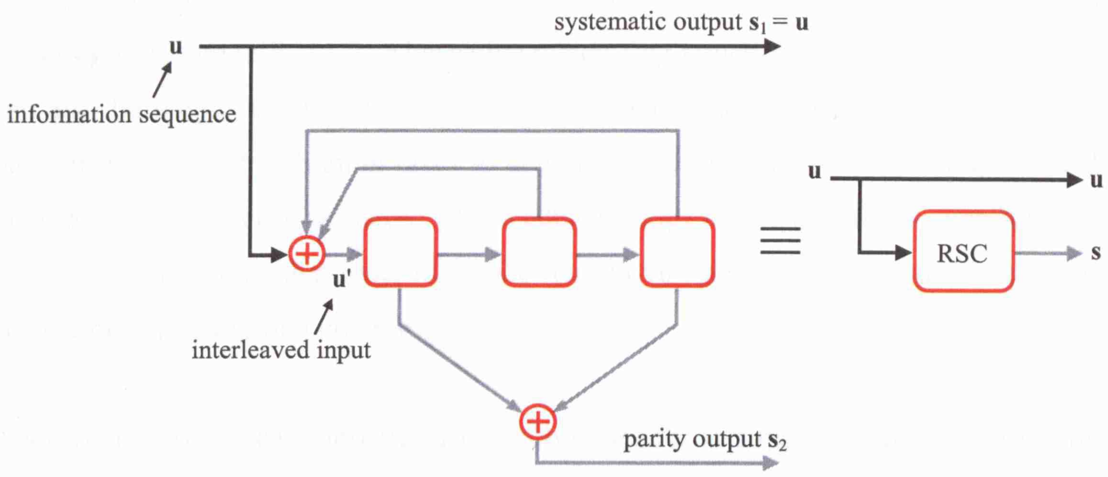


Fig. 2.11 A (2, 1, 3) RSC encoder constructed from the NSC encoder of fig. 2.6.

An RSC code can be constructed by a non-systematic convolutional (NSC) encoder as shown in fig. 2.11. One of the *n* outputs of the convolutional encoder of fig. 2.5 is fed back to the input and the information sequence is used in place of this output to make it systematic [69]. Fig. 2.11 shows the RSC version of the NSC encoder of fig. 2.6. Turbo codes utilise a parallel concatenation of typically two identical RSC encoders separated by a pseudo-random interleaver that has the purpose of rearranging the structure of the input sequence to the second encoder as depicted in fig. 2.12. This interleaver forms the basis of their powerful capability since it reduces significantly the possibility for the two RSC encoders to produce simultaneously an output of low Hamming weight [70]. This improvement is known and referred to as *interleaver gain*. The systematic outputs of the two RSC encoders (known as *constitute* encoders) shown in fig. 2.12 are omitted and an overall turbo encoder systematic output is taken straight from the input (**u**). Also, the rate of the turbo encoder without the output

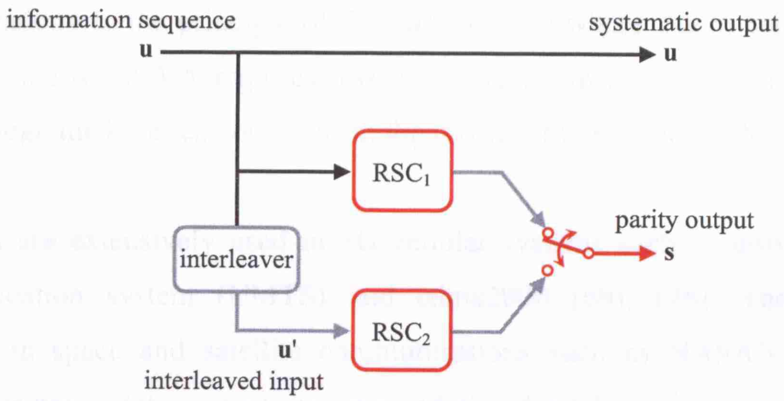


Fig. 2.12 A Turbo encoder constructed from two identical constitute RSC encoders of fig. 2.11.

switch depicted would be $R = 1/3$; however the switch provides puncturing effectively increasing the rate to $R = 1/2$. Alternative puncturing structures or no puncturing at all can also be employed in a turbo encoder. In practice, **s** and **u** are multiplexed in a parallel in/serial out fashion in order to generate the final codeword. It is important to mention that the recursive nature (the feedback loop) of the constitute encoders is a very important factor that determines the maximum interleaver gain, and hence, the correcting capability of a code [69].

Another important characteristic of the turbo encoder (a fundamental RSC property) is its *infinite impulse response* (IIR), explained as follows: An information input sequence of $H_w = 1$ produces an RSC parity output of infinite Hamming weight ($H_w \rightarrow \infty$) providing the initial condition of the registers is zero. However, in practice Turbo codes are used to encode k-bit input blocks (as in block codes, see fig 2.4) and in this case the register needs to return to state zero after the encoding of each successive input block. This process is known as *trellis termination* and it effectively converts the convolutional code to a block code. Since trellis termination for the overall block code using a turbo encoder can be rather complicated, mainly because of the interleaver between the two constitute encoders, a different representation of block codes has been devised known as *tail-biting* trellis to overcome this [71]-[72].

Decoding of Turbo codes is undertaken in an iterative manner using a soft input/soft output (SISO) processor such as *maximum a posteriori* (MAP) decoder [73] and *soft output Viterbi algorithm* (SOVA) [74]-[75]. The soft output of the first decoder feeds the second decoder (to form the first iteration) whose output is fed back a number of times before a decision is made to improve its performance [68]. This process can be viewed in analogy to the principle of the turbo engine where Turbo codes owe their name to. Generally, SOVA requires less hardware complexity to realise than MAP decoders though the latter can outperform the former at low SNR (E_b/N_o).

Turbo codes are extensively used in 3G cellular systems such as universal mobile telecommunication system (UMTS) and cdma2000 [69], [76]. They also find applications in space and satellite communications such as NASA's consultative committee for space data systems recommendation for telemetry processing in [77]. Finally, Turbo codes have started to replace convolutional codes in the concatenated coding scheme of digital video broadcasting (DVB) [78] with potential future adaptation and standardization.

CHAPTER 3

Implementing the VA and the MFDA

Until the late 1960's there were two major decoding techniques developed and used in convolutional codes namely *sequential* and *feedback decoding*. Sequential decoding was the earliest decoding method, proposed and analyzed by Wozencraft in 1957 [79] where using the *Fano* [80] or *stack algorithms* [81] it achieved a coding performance that asymptotically approached optimum. However, the computational complexity of this method substantially increased when realised to perform soft-input decisions, and in low SNR more computations were also necessary resulting in a slower operating speed. Feedback decoding [19] was another method developed later mainly for hard-decision decoding, taking advantage of the decoded output in order to reduce the computational complexity, resulting in sub-optimum coding performance. It essentially performed *sliding block* decoding using the tree diagram. The choice of the block depth L was of critical importance in determining the BER performance, and was shown to be dependent on the constraint length of the code [82]. Small values of L could result in error propagation that in turn could lead to catastrophic failure even if the code used was non-catastrophic. In 1967 the so-called *Viterbi algorithm* (VA) was presented by Viterbi [83] and later in 1973 exposed and clarified by Forney [20]. The VA theoretically performs *maximum likelihood* decoding [5] and although several decoding methods and modifications of existing techniques have since been proposed, it has been the most commonly employed decoding algorithm in convolutional codes.

Nowadays a Viterbi decoder (VD) can be realised using dynamic programming (software) or can be physically implemented in digital hardware [84]. Recently analogue realisations of its most functional blocks have been reported [11], [85]-[86] which allow a reduction in implementation complexity and power consumption, as it will further be discussed in chapter 4. Nevertheless, the need for digital path memory can not be eliminated, resulting only in minimal size reduction of the mixed-signal VD [11]. A more recent decoding method, the *modified feedback decoding algorithm* (MFDA), proposed in [18], can be realised almost entirely using analogue circuits, effectively reducing size and power dissipation. Although derived from the *feedback decoding algorithm* (FDA), it can be more closely described as a modification of the VA, since it employs its path elimination mechanism. Other convolutional decoding techniques can be found in [4] and [87].

This chapter is organised as follows: In section 3.1, a brief overview of the VA is discussed. In section 3.2, implementation considerations of the VD are reported in order to gain the appropriate background to present and evaluate the MFDA in section 3.3. In section 3.4, the *modified feedback decoder* (MFD) realisation is described using similar blocks as in the VD. The rest of the chapter is dedicated to various system-level simulations that compare their coding performance. Also, simulation results for the MFDA are given taking into account errors arising from the use of analogue components in practical convolutional decoders.

3.1 The Viterbi Algorithm (VA)

The VA uses the *likelihood function* $P(\mathbf{r}|\mathbf{s})$ to find the most probable transmitted sequence $\mathbf{s}$ using the received sequence $\mathbf{r}$. Its goal is to produce an accurate estimate $\hat{\mathbf{u}}$ of the information sequence. To do so it needs to find the sequence $\mathbf{s}'$ that maximises $P(\mathbf{r}|\mathbf{s})$, which is equivalent to minimising the probability of error. In the case where all possible transmitted sequences $\mathbf{s}$ are equally likely, the decoder is known to perform *maximum likelihood decoding* (MLD). Assuming transmission through a DMC the above statement can be expressed as

$$P(\mathbf{r} \mid \mathbf{s}') = \max P(\mathbf{r} \mid \mathbf{s}) \ \forall \mathbf{s} \tag{3.1}$$

and for a rate $R = k / n$ convolutional code

$$P(\mathbf{r} \mid \mathbf{s}) = \prod_{i=1}^{\infty} P(\mathbf{r}_i \mid \mathbf{s}_i) = \prod_{i=1}^{\infty}\prod_{p=1}^{n} P(\mathrm{r}_{pi} \mid \mathrm{s}_{pi}) \tag{3.2}$$

where r_{pi} is the pth bit of the ith received codeword of $\mathbf{r}$ and s_{pi} is the pth bit of the ith transmitted codeword of $\mathbf{s}$. Maximizing $P(\mathbf{r}|\mathbf{s})$ is equivalent to maximizing the logarithm $\log P(\mathbf{r}|\mathbf{s})$, known as the *log-likelihood* function [3], which effectively converts the product terms into summation terms and is expressed as

$$\log P(\mathbf{r} \mid \mathbf{s}) = \sum_{i=1}^{\infty} \log P(\mathbf{r}_i \mid \mathbf{s}_i) = \sum_{i=1}^{\infty}\sum_{p=1}^{n} \log P(\mathrm{r}_{pi} \mid \mathrm{s}_{pi}) \tag{3.3}$$

The VA needs to find the most likely path in a given trellis diagram, which is equivalent to maximising $\log P(\mathbf{r}|\mathbf{s})$ also referred to as *path metric Γ*. To do so, even for a finite codeword length L, it is necessary to compute Γ for 2^L different trellis paths and this becomes impractical for large L. However, taking advantage of the trellis structure the VA reduces this search to a limited number of paths known as *surviving paths* or *survivors*, discarding all paths that are highly unlikely. This is achieved using a technique known as *path elimination*, which significantly reduces the decoder complexity without degrading its performance. In this way, only $L{\cdot}2^{k(K-1)}$ path metrics need to be computed.

Observing the trellis diagram (see fig. 2.9) there are 2^k branches merging in each state after the first $K - 1$ levels. The VA performs path elimination in each state as follows: at each time instant it chooses and stores the path with the largest metric (survivor) and discards the remaining $2^k - 1$ paths; it also stores the metrics of the survivors such that at each time instant there are $2^{k(K-1)}$ surviving paths uniquely reaching each state, and $2^{k(K-1)}$ metrics associated with these paths stored for the next transition computations. Path elimination is the main computational core of the VA and it is performed by what is known as *add compare select* (ACS) operation, expressed by

$$\Gamma_{x,t+1} = \max\{\Gamma_{y,t} + \lambda_{yx,t}\} \quad \forall x, y : y = 1,2,\ldots,2^{k(K-1)} \tag{3.4}$$

where $\Gamma_{y,\,t}$ is the survivor path metric for state y at time instant t and $\lambda_{yx,\,t}$ is the *branch metric* resulting from the trellis transition from state y at time t to state x at time $t + 1$. The branch metric λ_i is denoted by the term $\log P(\mathbf{r}_i|\mathbf{s}_i)$ in equation (3.3) whereas $\log P(\mathrm{r}_{pi}|\mathrm{s}_{pi})$ refers to what is known as *symbol metric* μ_{pi}. Equation (3.4) shows the survivor path metric or *state metric* at $t + 1$ resulting from the addition of a preceding state metric and its corresponding transition branch metric in the trellis in order to reach state x. $\Gamma_{x,\,t+1}$ then needs to be stored and used in the next computational cycle. The ACS operation is further illustrated and explained in section 3.2.2.

In practice, since the information sequence and hence the received sequence $\mathbf{r}$ is of infinite length, it seems infeasible to realise the VA in order to incorporate hardware capable of storing an infinite number of survivors. Also the delay at the beginning of the decoding process will be infinite as the decoder outputs the first bit after the whole sequence has been processed. However, simulations have shown that using path elimination for a window depth of a few constraint lengths in the trellis diagram (typically $5K$ to $6K$), the decoded output associated with the oldest information bit is already known [88]. Further trellis steps (path eliminations) have no effect since all surviving paths emanate from the same one branch. Therefore, in practice the VA truncates $\mathbf{r}$ into a finite length $L = 5$ to $6K$, known as *decoding window depth*, which appears as an initial delay of L bits before the system outputs the first decoded bit; subsequently it decodes continuously at each trellis step. However, in this way the VA does not need to select the path with the largest metric and at the end of every decoding cycle (one trellis time step) it always retains one optimum path for each state. This means that the encoder state is unknown at each time instant, though providing an L of more than a few constraint lengths it is sufficient to determine the decoded output associated with the input to the encoder L steps before. In section 3.2 two alternatives for realising the VA will be considered that entirely depend on and determine the decoding window depth L.

The VA can be best explained with an example. Fig. 3.1 shows the hard-decision decoding steps for the encoder of fig. 2.6 including the survivors, branch metrics and path metrics at each stage. Note that for hard-decision decoding the path

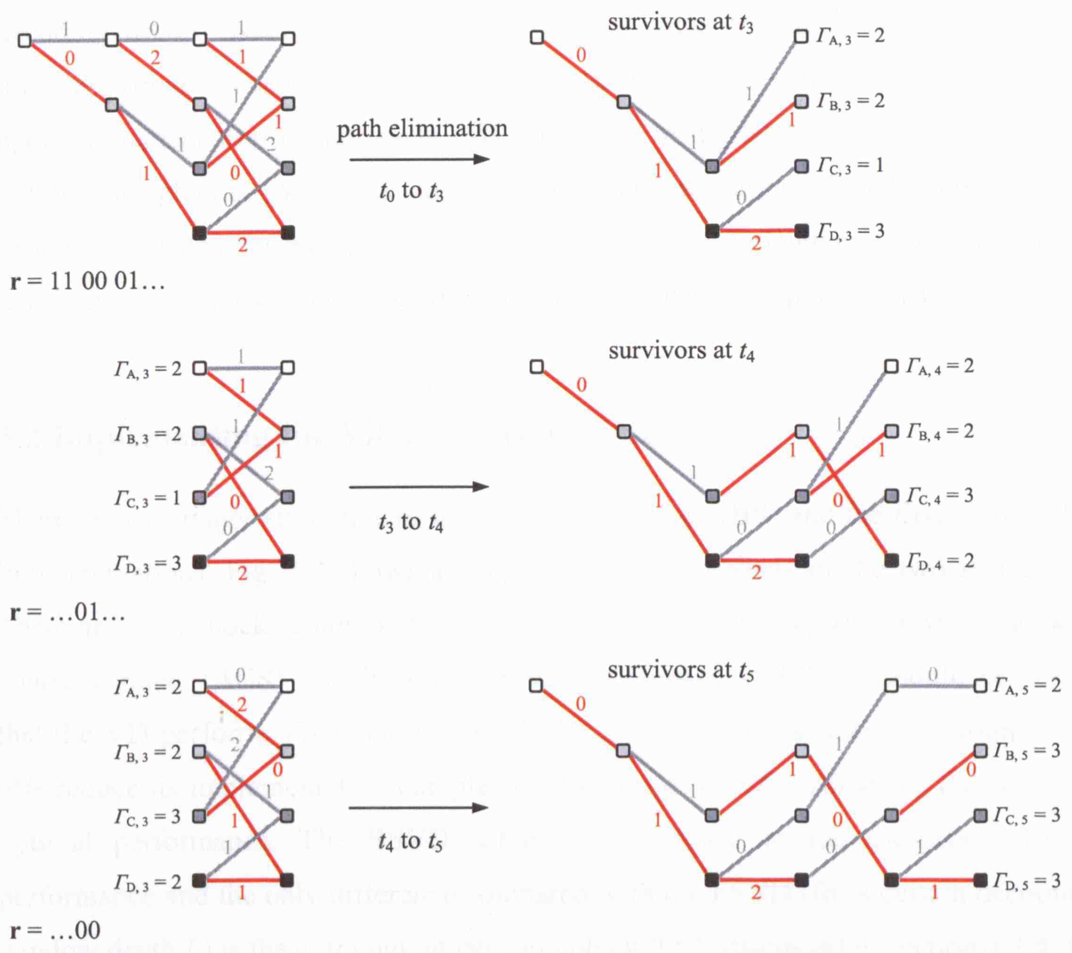


Fig. 3.1 Hard-decision Viterbi decoding steps as seen on the trellis for the encoder of fig. 2.6 assuming the codeword $\mathbf{s}$ = 11 10 00 01 10 and a received sequence $\mathbf{r}$ = 11 $\underline{0}$0 0$\underline{1}$ 01 $\underline{0}$0.

that minimises the Hamming distance between $\mathbf{s}$ and $\mathbf{r}$ should be chosen, as discussed in section 2.1.3. Therefore, in this case, maximising the log-likelihood function of equation (3.3) is equivalent to minimising the path metric. However, in order to retain the term maximum in the ACS operation while at the same time refer to Γ as the log-likelihood function $\Gamma = \log P(\mathbf{r}|\mathbf{s})$, the branch metric λ_i is inverted and the maximum λ_i is equivalent to the minimum Hamming or Euclidian distance.

It can be shown that soft-decision Viterbi decoding using binary phase shift keying BPSK modulation over an AWGN channel can favour the BER performance between about 2 to 3 dB in terms of CG over its hard-decision version [24]. Simulations show that an improvement in CG of about 2dB is more realistic [88] (in practical VDs of short constraint length $K \leq 9$), increasing with E_b/N_o asymptotically to the value of 3dB. Also, an 8-level (3-bit) quantisation in the demodulator output of soft-decision decoding usually results in a trivial CG loss of about 0.25dB compared with the infinite-level quantisation (or no quantisation). This is very important in deciding upon implementation design trade offs especially in digital realisations of the VD. Finally, in practice VDs are restricted to short constraint lengths due to the exponential dependence of computational complexity on *K*. However this is adequate to achieve moderate decoding performance at high data rates up to a few Gb/s.

3.2 Implementing the Viterbi Decoder (VD)

There are two major structures of the VD: the *best-state* (BS) and the *fixed-state* (FS) implementations. Fig. 3.2 shows a simplified block diagrams of the two structures. There are three blocks common to both: the *branch metric computer* (BMC), the *add compare select* (ACS) and the *survivor storage memory* (SSM). It should be noted that the VD performs *theoretically only* MLD as common practical realisation trade offs reduce its implementation complexity (hardware or software) at the cost of sub-optimal performance. The BSVD achieves the closest to the theoretical MLD performance and the only difference compared with the FSVD (for a certain decoding window depth *L*) is the extra output decision block (OD) discussed in section 3.3.4. In the following sections the function of these blocks will be briefly described with more emphasis on the BMC and the ACS blocks which will form the basis for the development and analysis of the MFD.

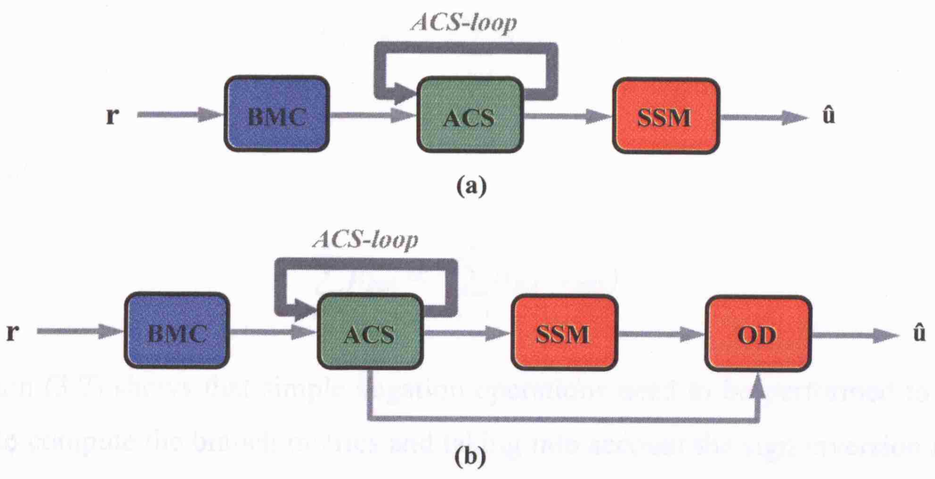


Fig. 3.2 Block diagram of a **(a)** fixed-state VD (FSVD) and **(b)** best-state VD (BSVD)

3.2.1 Branch Metric Computer (BMC)

The BMC is responsible for all the branch metric computations using a suitable distance metric. At each time instant there are 2^{kK} branches associated with a trellis transition for which 2^n different branch metrics need to be produced. For optimum soft-decision Viterbi decoding over an AWGN channel the preferred metric is squared Euclidian distance between each n-bit received block $\mathbf{r}_t$ at time t and its 2^{kK} associated n-bit branch words $\mathbf{s}_{yx}$, representing all possible transitions in the trellis from time t to $t+1$. This is expressed as

$$\lambda_{yx,t} = (\mathbf{r}_t - \mathbf{s}_{yx})^2 = \sum_{p=1}^{n} (\mathrm{r}_{p,t} - \mathrm{s}_{pyx})^2 = \sum_{p=1}^{n} \mu_{pyx} \tag{3.5}$$

where μ_{pyx} represents the bit metric for the transition from state y at t to state x at $t+1$. Taking into consideration that adding or subtracting a constant from all branch metrics $\lambda_{yx,t}$ simultaneously has no effect on the performance of a MLD, equation (3.5) can be simplified to

$$\lambda_{yx,t} \propto \sum_{p=1}^{n} (\mathrm{s}_{pyx})^2 - 2\sum_{p=1}^{n} (\mathrm{r}_{p,t} \cdot \mathrm{s}_{pyx}) \tag{3.6}$$

Also, since BPSK is the modulation scheme for optimum VD performance [88], the modulator assigns one value to each code bit from an *antipodal signal* set $(-a, +a)$. In the case where $a = 1$, an s_{pyx} of binary “0” is assigned to -1 and an s_{pyx} of binary “1” to $+1$ and this can further simplify (3.6) to

$$\lambda_{yx,t} \propto 1 - 2\sum_{p=1}^{n} (\mathrm{r}_{p,t} \cdot \mathrm{s}_{pyx})$$

and hence,

$$\sum_{p=1}^{n} \mu_{pyx} \propto -\sum_{p=1}^{n} (\mathrm{r}_{p,t} \cdot \mathrm{s}_{pyx}) \tag{3.7}$$

Equation (3.7) shows that simple negation operations need to be performed to $\mathrm{r}_{p,\,t}$ in order to compute the branch metrics and taking into account the sign inversion in $\lambda_{yx,\,t}$ discussed in section 3.1, this is illustrated more clearly for each bit metric in

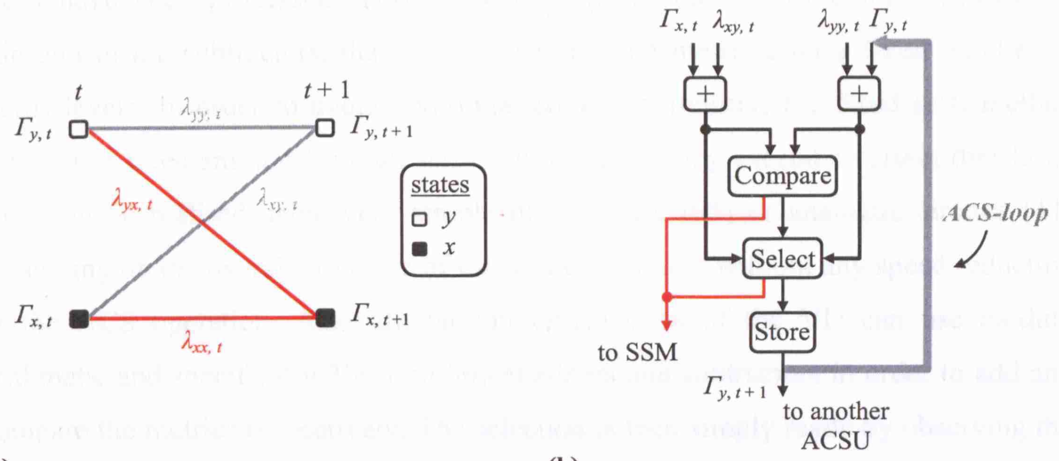
$$\mu_{pyx} \propto \mathrm{r}_{p,t} \cdot \mathrm{s}_{pyx} = \begin{cases} \mathrm{r}_{p,t} & \text{for } \mathrm{s}_{pyx} = +1 \\ -\mathrm{r}_{p,t} & \text{for } \mathrm{s}_{pyx} = -1 \end{cases} \qquad 1 \le p \le n \tag{3.8}$$

Finally, for a hard-decision VD over a BSC the preferred metric is Hamming distance (in this case inverse Hamming distance) expressed as

$$\mu_{pyx} = \begin{cases} 1 & \text{when } \mathrm{s}_{pyx} = \mathrm{r}_{p,t} \\ 0 & \text{when } \mathrm{s}_{pyx} \neq \mathrm{r}_{p,t} \end{cases} \qquad 1 \le p \le n \tag{3.9}$$

3.2.2 Add Compare Select (ACS)

The ACS block performs path elimination realising equation (3.4) using the branch metrics computed by BMC while it also updates the SSM block accordingly. The trellis diagram can be divided into a number of 2-state basic modules where for an $R = 1/n$ convolutional code each module, known as "butterfly", has the form of that shown in fig. 3.3a. The ACS unit (ACSU) for a state y adds at each time instant the branch metric $\lambda_{yy,\,t}$ to the state metric $\Gamma_{y,\,t}$, and $\lambda_{xy,\,t}$ to $\Gamma_{x,\,t}$ and selects the maximum candidate to form the path metric $\Gamma_{x,\,t+1}$; it then stores that metric for one time increment in order to be fed-back and used as the state metric for the next ACS computations, as depicted in fig. 3.3b. It also needs to update the path memory SSM according to the information bit associated with the selected branch.



(a) **(b)**

Fig. 3.3 (a) 2-state "butterfly" module of an $R = 1/n$ convolutional code trellis representation, **(b)** functional block diagram of the ACSU.

Note that the time it takes for one ACSU to add, compare and store the selected path metric, defined in this context as the ACS *feedback-loop delay*, is of critical importance; it primarily determines the decoding cycle (operational speed) of the VD in a *state-parallel* configuration [89], where $2^{k(K-1)}$ identical ACSUs (one for each state) are used in parallel at the cost of increased complexity and size [10]. This is often the case in high speed VDs where the sequential use of one ACSU unit in a *state-serial* configuration is not affordable in terms of speed degradation [90]. Area-efficient methods have been devised combining the merits of both configurations and effectively trade speed for area [91]-[92]. Also, *pipelined* architectures can further improve the speed limitations imposed by the ACS block in an area-efficient VD without significant increase in complexity [7], [93]. In general, a suitable ACS architecture should be adopted for optimum performance depending on system specifications and requirements.

The optimum VD performs path elimination for an infinite number of trellis steps (as long as the information sequence) and this imposes a major constraint concerning the feasibility of either digital or analogue circuit realisations that can accommodate for the infinite growth of path metrics. Several methods have been proposed to overcome this phenomenon, known as *path metric overflow*. These include the *variable shift*, *fixed* or *threshold shift* and *modulo normalization* techniques [94]. The variable shift subtracts the minimum state metric from all metrics after a certain number of trellis steps have been processed. This however requires the use of extra hardware, in addition to the subtractors, that select the minimum metric after a fixed number of trellis levels. In order to avoid additional complex circuitry, the fixed shift method subtracts a fixed amount from all state metrics when they exceed a certain threshold. This can be realised using very simple digital logic [95] or analogue circuits [11] depending on the overall structure of the decoder, almost without any speed reduction in the ACS operation. Also, digital implementations of the VD can use modulo arithmetic and specifically 2's compliment adders and subtractors in order to add and compare the metrics respectively. The selection is then simply made by observing the sign bit (carry) [94].

3.2.3 Survivor Storage Memory (SSM)

The SSM block is used to store the surviving paths in a hypothetical digital representation. It records the history of the ACS decisions for a decoding window depth L (L trellis transitions) which in practice is chosen to be between $5K$ and $6K$ in a FS implementation of the VD. There are mainly two memory organisation strategies that can be employed in order to realise the SSM. Both require a certain amount of 1-bit registers (e.g. D-type flip-flops) proportional to L; these are the *register exchange* and *traceback* methods. In the following, a general $1/n$ convolutional code will be considered for simplicity, without loss of generality, in order to illustrate these two methods.

The register exchange [96] necessitates the use of $L{\cdot}2^{(K-1)}$ registers connected together in a structure that mimics the trellis connections in each successive stage. That is, it uses $2^{(K-1)}$ shift registers, one for each state, capable of storing L bits. Their interchange ability is determined by the survivors at each stage by means of multiplexers; each multiplexer is controlled by the information bit associated with the corresponding ACS decision. Fig. 3.4 illustrates the register exchange configuration for the trellis diagram of fig. 2.9. The output of each L-bit shift register for all 4 states reflects the decoded output related to the information bit L stages back in the trellis.

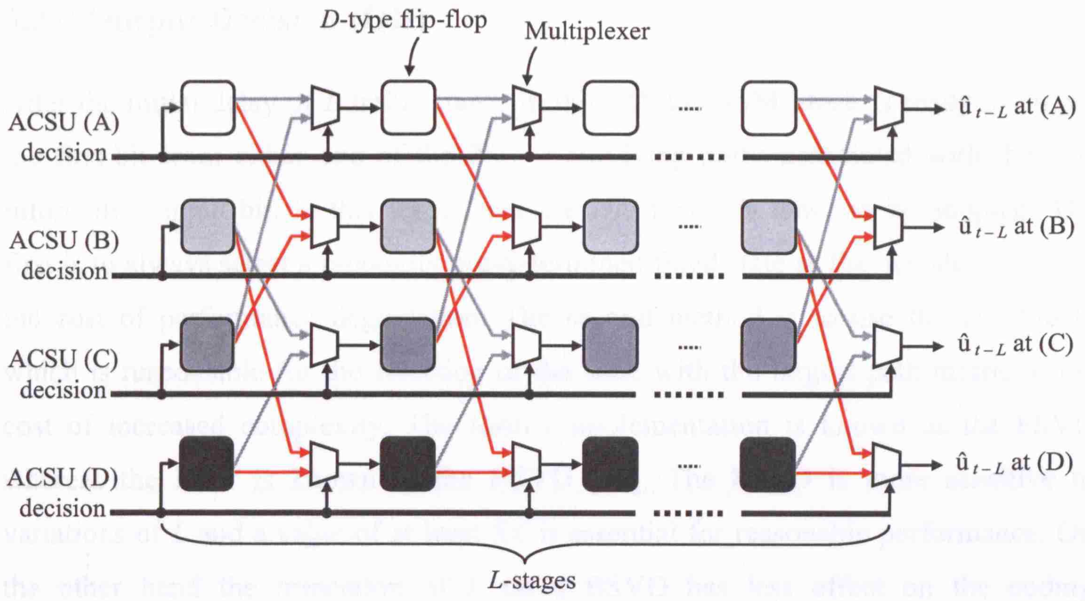


Fig. 3.4 Register exchange memory organisation in the VD for the code of fig. 2.9.

Providing a large $L > 5K$, the contents of the last flip-flop in each register should almost always be the same and equal to u_{t-L}. Although this technique is straight forward since it is directly derived from the decoder's requirements, it involves a large amount of area necessary to store the complete history of all surviving paths at each time instance. This can also lead into unnecessary power consumption due to its implementation complexity, especially at high speed.

In the trace-back method [97] only the recursive decisions of the ACS need to be stored while an extra mechanism is required that uses *pointers* to trace these decisions in reverse; recalling these pointers after an initial delay is equivalent to reproducing the $2^{k(K-1)}$ surviving paths. The trace-back memory is organised as a cyclic buffer and is partitioned into *read* and *write* regions that run in parallel. As a new ACS decision sequence is inserted in the write region, the surviving path is traced and read out from the read region in each state [98]; the first bit stored in the read region corresponds to the information bit prior to L stages in the trellis. Each trace-back can elaborate more than a single decoded output depending on the architecture employed. This technique is beneficial compared to the register exchange in terms of size and power dissipation especially in higher order codes (e.g. $K > 5$), but it has an increased delay caused by the trace-back mechanism, and is therefore suitable for VDs of moderate data rates (< 10MHz).

3.2.4 Output Decision (OD)

After the initial delay of L trellis stages in the VD, the SSM block is ready to output the first bit from either one of the $2^{(K-1)}$ surviving paths associated with the first information input bit. In this stage there are two methods that can be adopted. The first is to always select a randomly pre-determined fixed state as the decoder output at the cost of performance degradation. The second method is to use the OD block, which is responsible for the selection of the state with the largest path metric at the cost of increased complexity. The former implementation is known as the FSVD whereas the latter is known as the BSVD [99]. The FSVD is more sensitive to variations of L and a value of at least $5K$ is essential for reasonable performance. On the other hand the truncation of L on a BSVD has less effect on the coding performance (see section 3.5.1) and the decoding window depth can be further reduced to values lower than $4K$ without significant CG loss. This reduction in memory size in the BSVD is very important especially in analogue VD realisations, considering that the SSM block can occupy up to as much as 50% of the total die area [11]. However, the required OD block, which is usually a *winner take all* (WTA) network [100]-[101], imposes implementation limitations since its hardware complexity increases exponentially with K and therefore is not feasible for $K > 5$. In both cases with or without the OD block, an optimum solution can be determined by exhaustive simulations, trading coding performance against implementation complexity and vice versa until desired requirements are met.

3.3 The Modified Feedback Decoding Algorithm (MFDA)

The MFDA is an area efficient method that eliminates the extra digital storage block (SSM) required in a VD [18]. It was essentially derived from a closer observation of the combined advantages of FDA and VA. In the FDA, the feedback operation is used to determine the initial branching of the new sub-tree in each decoding cycle, and there is no need for SSM as there is no path elimination. Instead, a look-up table providing all possible paths within the sub-tree (determined by the sliding block depth) is used to establish the best match for the received sequence **r**. However, in order to prevent catastrophic failure, the depth L of the sliding block should be at least

$$L \geq \left\lfloor \frac{K}{2} \right\rfloor + 1 \tag{3.10}$$

for any $R = 1/2$ non-catastrophic convolutional code [82]. Also, simulations show that a feedback decoder (FD) should have an L of at least a few constraint lengths in order to achieve asymptotically the coding performance of a VD. This imposes severe implementation limitations, considering that the size of the look-up table grows exponentially with L, restricting the realisation of the FD to only small codes ($K < 5$). This is where the advantages of path elimination employed by the VA underlie the origin of the MFDA. In the MFDA, after the initial branching is determined (based on previous feedback decisions), path elimination then takes place for the rest of the sliding block depth, in order to compensate for possible propagation errors. In this way, a feedback decoder using a decoding depth of $L = 5K$ to $6K$ (as in a FSVD) becomes feasible. The size of the look-up table is independent of L, as it is restrained to cover only the first K branches of the tree. At every decoding cycle the look-up table is updated by $K - 1$ previous outputs. However, as there is no need to track and trace the most likely path but only preserve the identity of its origin (upper or lower sub-tree as in the FDA), the SSM is effectively eliminated.

The proper operation of the MFDA can best be illustrated using the *modified tree* principle shown in fig. 3.5. The MFDA splits the tree diagram of fig. 2.8 into two sub-trees (upper and lower) converting them effectively into trellises; it then performs path elimination like the VA in both sub-trellises individually and simultaneously. Assuming a sliding block (decoding window depth) of L levels, the first $K + 1$ levels are processed in parallel (including the first path elimination) and the remaining $L - K - 1$ are processed sequentially. In particular, all branches associated with the first $K - 1$ levels in the modified tree need to be updated by $K - 1$ previous decoded outputs in

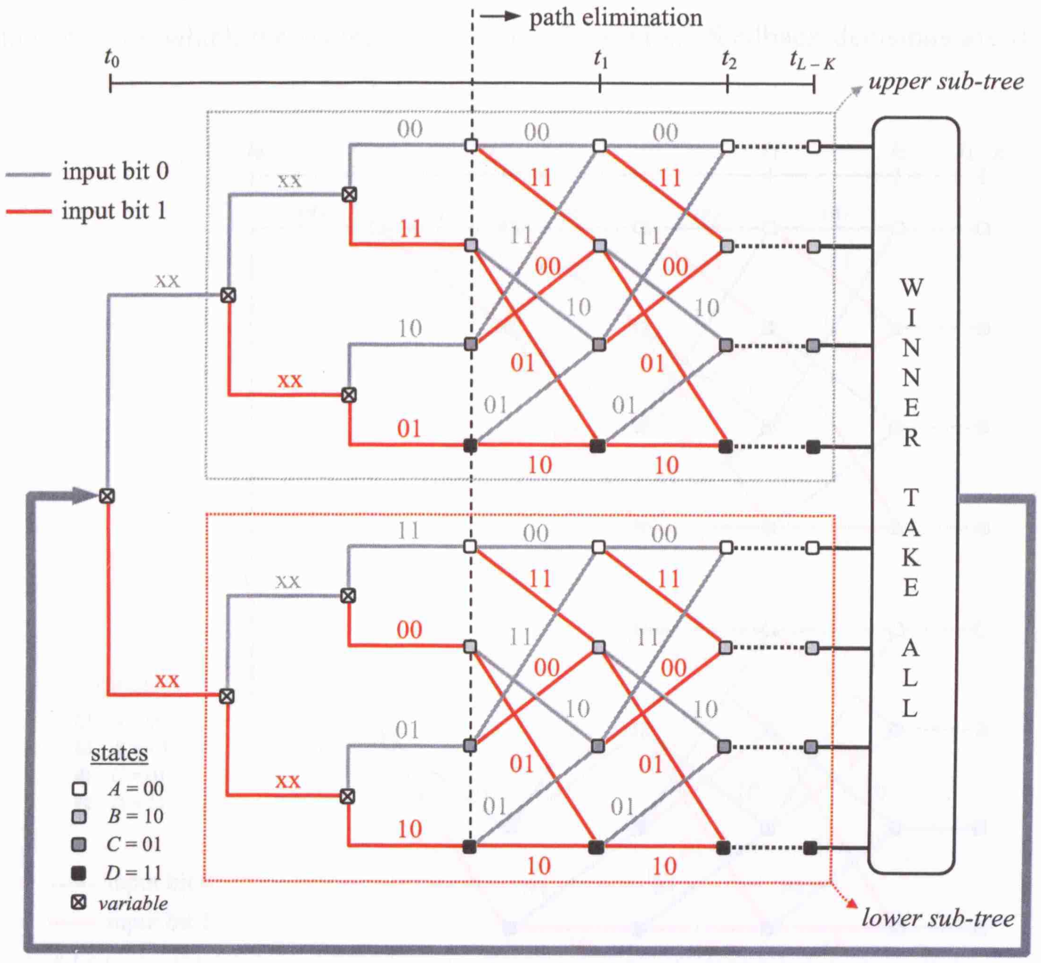


Fig. 3.5 The modified tree principle of operation employed by the MFDA for the encoder of fig. 2.6.

order to determine the beginning state of the new sliding block in both sub-trees and return them to a known state. This means that all states associated with the first $K-1$ levels are variable, whereas in the remaining block (of depth $L-K+1$) all states are fixed, as depicted in fig. 3.5. When all L levels of a sliding block are processed, the MFDA simply needs to identify whether the path with the largest metric lies in the upper or lower sub-trellis, resulting in a decoded output of "0" or "1", respectively. As path elimination is performed in exactly the same manner as in the VA, at the end of every decoding cycle there are 2^{K-1} surviving paths in each sub-trellis, and therefore the comparison extends to a total of 2^K different path metrics.

An example of a hard-decision MFDA in decoding the received sequence $\mathbf{r}$ = [11 10 00 01] will follow to visually demonstrate its principles of operation. This sequence results directly from the noiseless transmission of the codeword corresponding to the information sequence $\mathbf{u}$ = [1 1 0 1] (see fig. 2.8). Assuming the decoder is initially in state A = 00, which means that the $K-1 = 2$ previous feedback decisions are 0, the

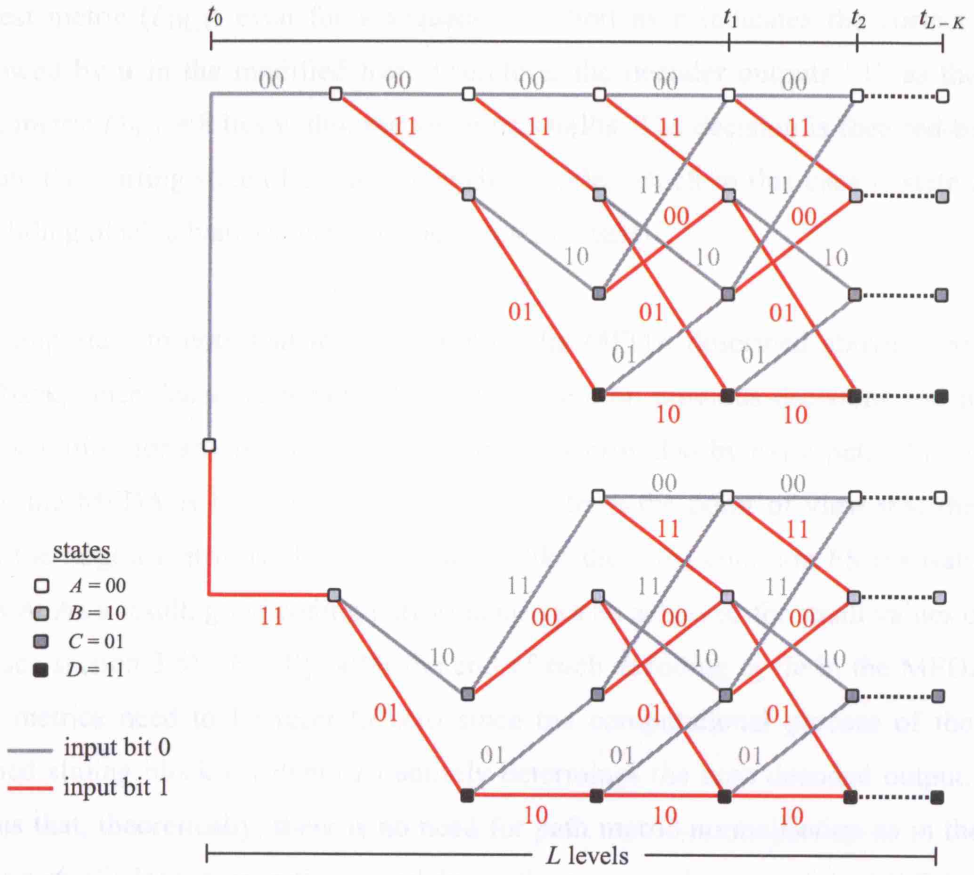


Fig. 3.6 The modified tree of fig. 3.5 when the initial state of the decoder is A.

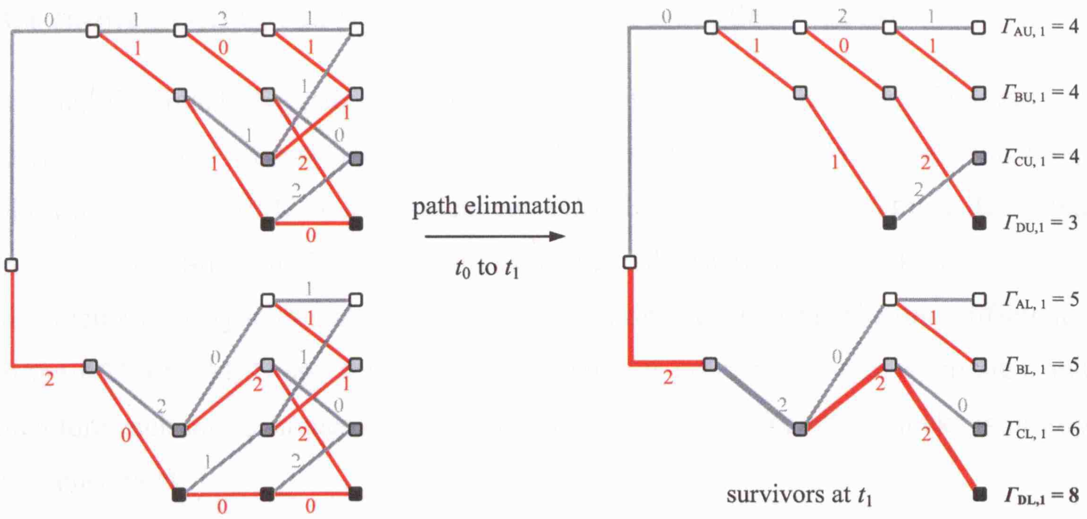


Fig. 3.7 Hard-decision decoding of the **r** = [11 10 00 01] employing the MFDA, assuming **r** = **s** (i.e. noise free transmission).

modified tree has the form shown in fig. 3.6. Employing the inverse Hamming distance (in order to always select the path with the largest metric), the surviving paths and their metrics after the first path elimination at t_1 for both sub-trellises are shown in fig. 3.7. As there is no noise corrupting the codeword, the path with the largest metric ($\Gamma_{DL,1}$) even for a sequence as short as **r** indicates the correct route followed by **u** in the modified tree. Therefore, the decoder outputs "1" as the path with metric $\Gamma_{DL,1} = 8$ lies within the lower sub-trellis. The decision is then fed-back to update the starting state of the new decoding cycle, which in this case is state B and the sliding block advances one step ahead (on the right).

It is important to note that the operation of the MFDA described above is *pseudo-*feedback, since decoding is not entirely dependent on previous decisions and further compensation for any possible fed-back errors is provided by extra path elimination. Also, the MFDA is truly maximum likelihood from the point of view that the path with the largest metric is always selected unlike the most common FS realisation of the VA. As a result, good coding performance can be achieved for small values of $L < 4K$ (see section 3.5). Finally, after the end of each decoding cycle in the MFDA the path metrics need to be reset to zero since the computational process of the new applied sliding block (of depth L) entirely determines the next decoded output. This means that, theoretically, there is no need for path metric normalisation as in the VA at the cost of a loss in operating speed due to the sequential nature of the MFDA.

3.4 Implementing the Modified Feedback Decoder (MFD)

A simplified block diagram of a MFD is shown in fig. 3.8. It consists of the *symbol storage* (SS) and *winner take all* (WTA) blocks that effectively replace the SSM block of a VD, the BMC and two identical ACS blocks very similar to those described in section 3.2. Since the MFD is mainly a sequential system some clocking scheme is also required for synchronisation purposes (not shown in fig. 3.8). The only difference in the ACS of a MFD and a VD is that here there is no memory update circuitry and therefore each unit becomes less complex. The function of the other blocks is briefly explained next.

3.4.1 Symbol Storage (SS)

Since the received sequence **r** is of infinite length, this block is necessary to buffer L samples of **r** at each decoding cycle. These samples are processed by the BMC in order to update the ACS blocks according to one sliding block information. After the corresponding decision of the WTA, the oldest sample stored in the SS is shifted out while a new sample is inserted in a serial-in/serial-out fashion and the sliding block advances one step ahead. Hence, the SS consists of n analogue delay lines, one for each code bit. n simple sample and hold circuits of L stages each can be used. The contents of the last $n(K + 1)$ stages are processed in parallel and the remaining $n(L - K - 1)$ are processed sequentially.

3.4.2 Branch Metric Computer (BMC)

The BMC is slightly different from that described in section 3.2.1. Its function is to calculate the branch metrics for all possible paths in a modified tree which is identical

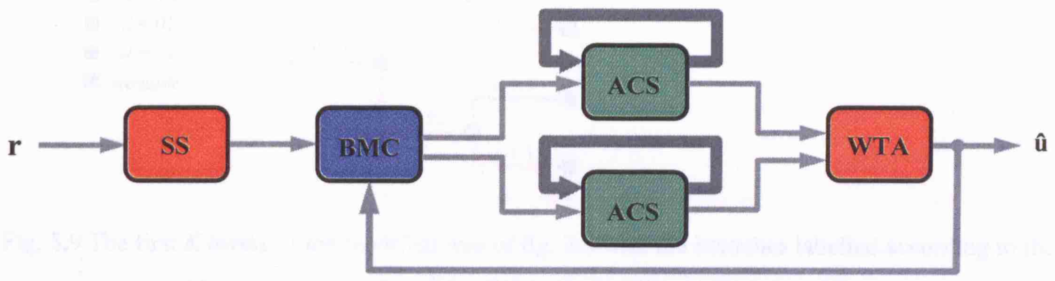


Fig. 3.8 Block diagram of a MFD.

to the BMC of a VD; however, the variable states corresponding to the first K levels of every sliding block (as shown in fig. 3.5) require updating at the beginning of each decoding cycle. For illustration purposes regarding the dependence of these levels on $(K-1)$ previous decoded outputs, the first K levels of the modified tree of fig. 3.5 are redrawn in fig. 3.9. Here, each branch is labelled with the contents of the encoder corresponding to that transition, where $\hat{u}_{t-1}$ and $\hat{u}_{t-2}$ are the previous decoded bit and the one before that, respectively. In this case two previous decoded outputs are necessary and used to return the decoder to a known state before the sequential operation of path elimination begins. From a closer observation of the modified tree, it can be seen that at any state the transition code bits corresponding to an input of 0 are the inverted transition code bits corresponding to an input of one at the same state. Therefore, the extra combinatorial logic used in the BMC in order to produce the first $2 + 2^2 = 6$ variable branch metrics is limited to the calculation of only $1 + 2 = 3$ metrics as the remaining metrics are derived by simple inversions. In general, for any $(n, 1, K)$ convolutional encoder there are:

$$\text{number of variable branches} = \sum_{i=1}^{K-1} 2^i \tag{3.11}$$

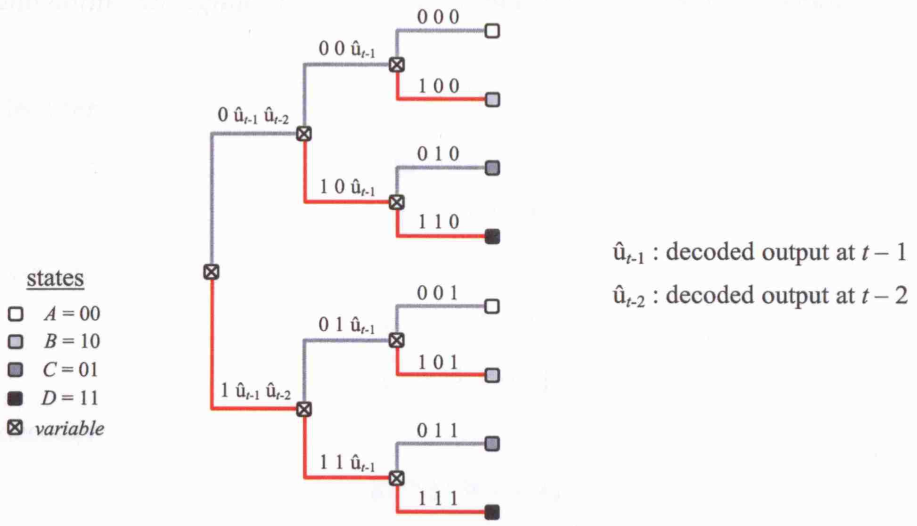


Fig. 3.9 The first K levels of the modified tree of fig. 3.5 with the branches labelled according to the contents of the encoder of fig. 2.6 corresponding to these transitions.

3.4.3 Winner Take All (WTA)

The WTA block is the most crucial block in a MFD since it determines its decoded output. After all L levels in a sliding window have been processed there are 2^{K-1} path metrics in each sub-trellis (upper or lower). The WTA is responsible for identifying the initial branching of the path with the largest metric, comparing 2^K survivors at the end of every decoding cycle. It therefore outputs "0" if the winning path lies within the upper sub-trellis or "1" otherwise. Different structures of WTA networks can be used effectively trading implementation complexity against coding performance.

3.5 System-level Simulations and Comparisons

In order to investigate the coding performance of the MFDA, an encoder-decoder model was implemented using Simulink® blocks in Matlab environment. These models were converted into *C* code using the Real Time Workshop. This enabled the simulation of complete systems up to and including $K = 5$ codes. Further extrapolation of the simulation results may extend drawn conclusions to include decoders of higher values of K. Analogue implementation issues in the MFD are considered and simulation results compare its BER performance to that of the VD. The following optimised $R = 1/2$ convolutional codes were investigated using *baseband antipodal signalling* or *bipolar signalling* in an AWGN channel:

$K = 3$ decoder:

$$\mathbf{g}_1 = [1\ 0\ 1]$$
$$\mathbf{g}_2 = [1\ 1\ 1]$$

$K = 4$ decoder: (3.12)

$$\mathbf{g}_1 = [1\ 1\ 1\ 1]$$
$$\mathbf{g}_2 = [1\ 1\ 0\ 1]$$

$K = 5$ decoder:

$$\mathbf{g}_1 = [1\ 0\ 1\ 1\ 1]$$
$$\mathbf{g}_2 = [1\ 1\ 0\ 0\ 1]$$

where $\mathbf{g}_1$ and $\mathbf{g}_2$ are the generator polynomials for each code. Fig. 3.10 depicts the Simulink® model used for the simulations of various MFDs and VDs.

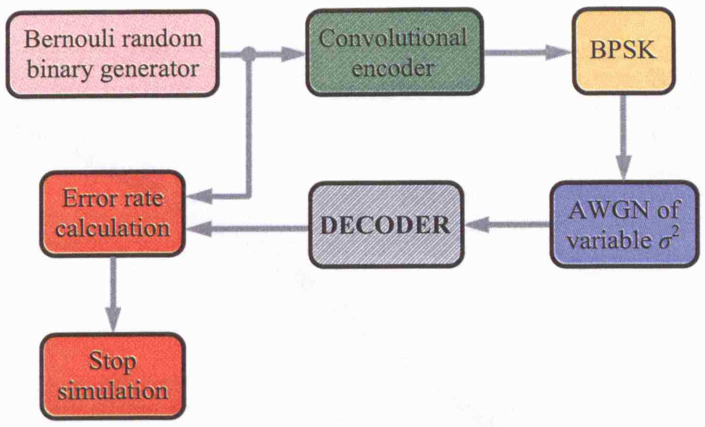


Fig. 3.10 Simulink® simulations modelling.

3.5.1 Establishing an optimum *L* for the MFD

The decoding depth L is of significant importance in a practical realisation of the VA, determining its coding performance and implementation complexity. A safe choice of $L = 6 \cdot K$ in a FSVD, results in a loss of CG of about 0.25dB compared to the theoretical $L \rightarrow \infty$ [102]. However, simulations show that similar performance can be achieved by the BSVD truncated to a smaller L of about $3K$ to $4K$, at the cost of increased complexity. Figures 3.11 to 3.13 show plots of *BER* against E_b/N_o for various values of L (MFD and BSVD); the FSVD with $L = 6K$ is also included in the graphs. As these systems were firstly investigated assuming ideal circuit operation, modelled by ideal Simulink® blocks, the input to the decoders was neither quantised nor limited assuming infinite resolution. From these plots it can be seen that the same BER performance as that of a FSVD can be achieved by the MFD but for smaller values of L ($L < 4K$). This has a direct impact on the decoder design trade offs showing that a MFD can be constructed using low complexity circuits, and effectively, compete even with a BSVD in terms of CG. Also, as it will be shown in section 3.5.3, shorter L means less sensitive system to analogue implementation errors encountered in practice. Therefore, the optimum L for the MFD lies, generally, within $3K$ and $4K$ and is more accurately tuned depending on the size of the decoder (K) and the specific circuitry, which in turn determine the error conditions under which decoding operation is guaranteed. For purposes of generic investigation of the behaviour of the MFD, the following values of L are chosen: $L = 11$ in a $K = 3$, $L = 16$ in a $K = 4$ and $L = 20$ in a $K = 5$ systems.

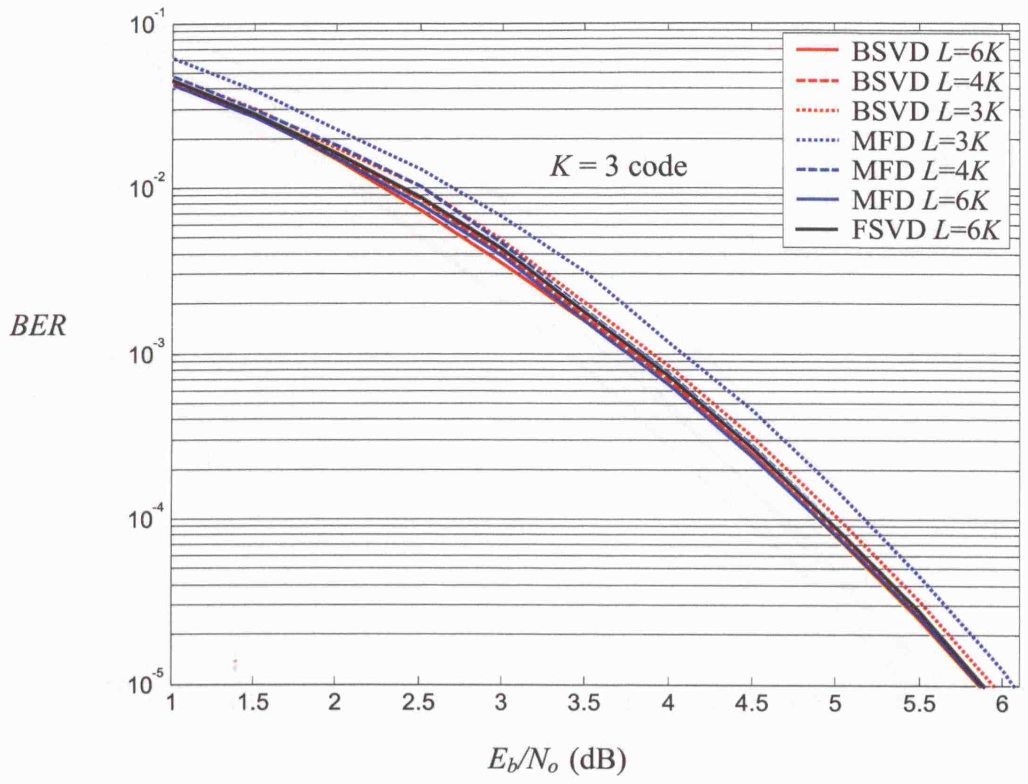


Fig. 3.11 Coding performance of the MFD and the VD for $K = 3$ code and various L.

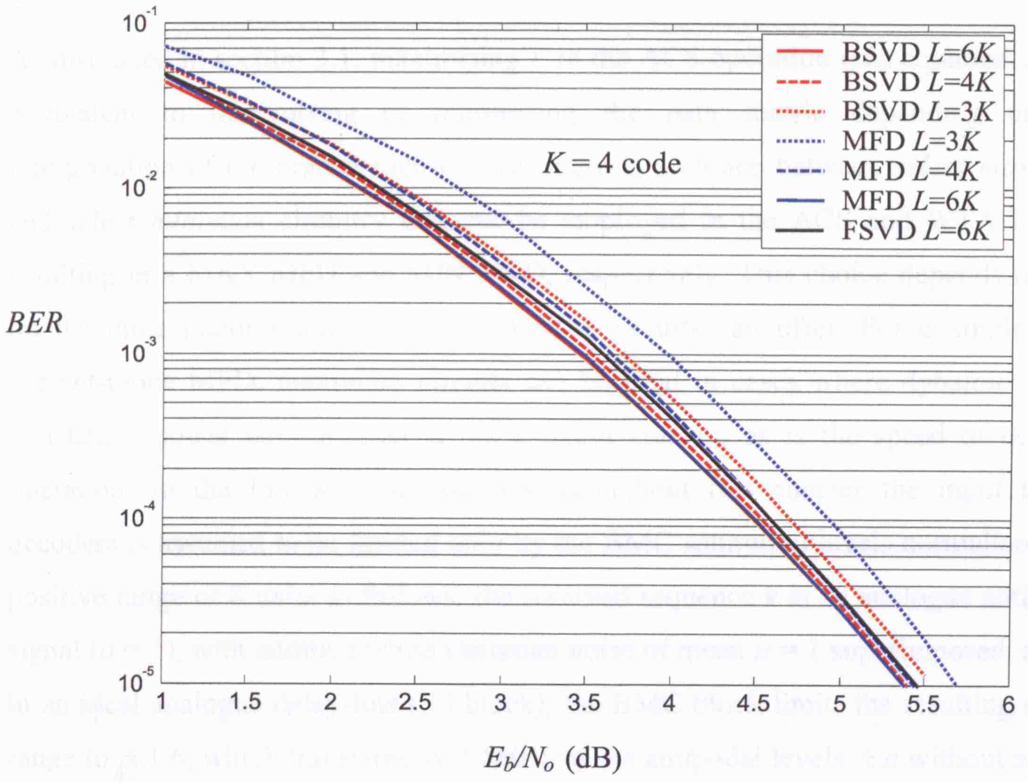


Fig. 3.12 Coding performance of the MFD and the VD for $K = 4$ code and various L.

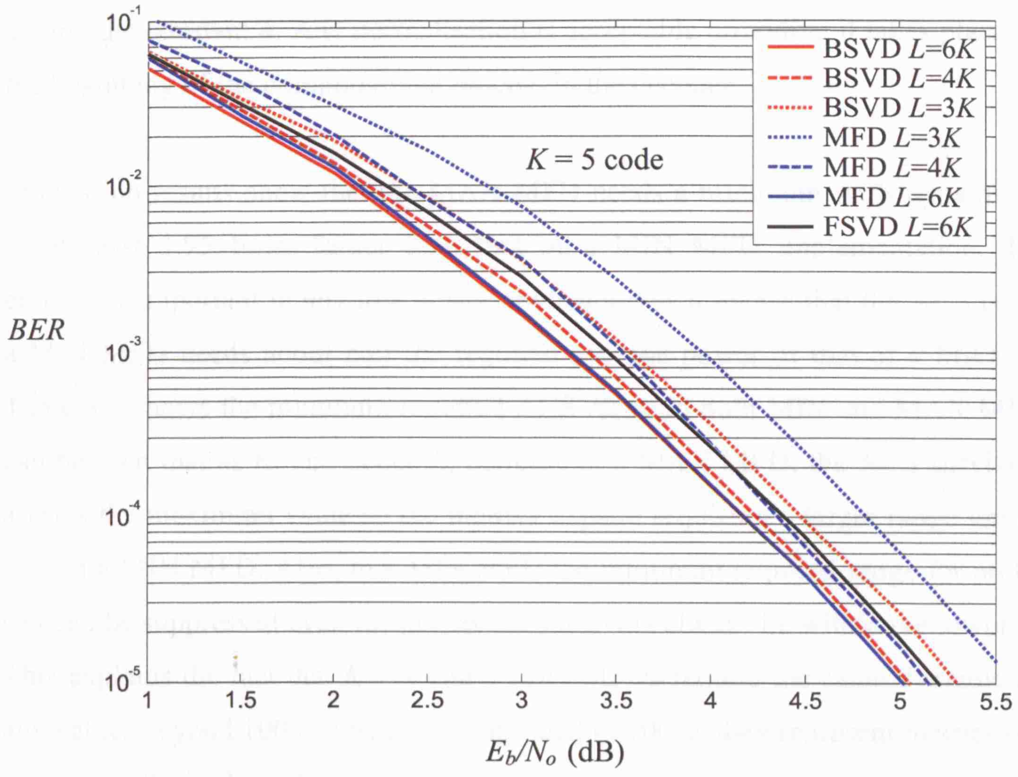


Fig. 3.13 Coding performance of the MFD and the VD for K = 5 code and various L.

3.5.2 MFD MIN/MAX Configuration Comparisons

As discussed in section 3.1, maximising Γ in the ACS operation (see equation 3.4) is equivalent to maximising or minimising the path metric depending on the interpretation of the branch metric λ. So there is a choice between *select maximum* and *select minimum* circuitry that can be employed in the ACS and WTA blocks, resulting in a MAX MFD and MIN MFD, respectively. This choice depends mainly on the three major trade offs that dedicated circuits can offer. For example, in a current-mode MFD, maximum circuits can be used in cases where dynamic range, and hence power consumption is not a major concern as is the speed of decoder operation. In the following simulations throughout this chapter the input to the decoders is assumed to be limited only by the BMC saturation levels normalised to a positive range of 8 units as follows: the received sequence **r** is an analogue antipodal signal (a = 1), with additive white Gaussian noise of mean μ = 1 superimposed, stored in an ideal analogue delay-line (SS block); the BMC block limits the resulting signal range to ± 1.6, which translates to ± 160% of the antipodal levels ± a without almost any loss in CG. This is then normalised to a dynamic range of 0 to 8 *units* (in practice a unit can be mV or μA) by multiplying the received signal by a factor of 2.5 and adding the constant 4. Any normalisation is acceptable providing it takes place before the beginning of the computational process in the decoder.

Simulation results show that the MAX MFD needs a minimum ACS dynamic range of at least 1.95 times larger than that of a MIN MFD implementation. This is extremely important in any low power application as it means that the ACS circuit in a MIN MFD needs about half the required average power of that of a MAX MFD. Table 3.1 shows the minimum required ACS range in both MIN and MAX MFDs. It can be seen that as K, and hence L, increase in a MAX MFD, the ACS survivors are always the maximum value so the metrics expand requiring a larger range unlike the case in a MIN MFD. Also, in a MIN MFD the minimum required range for no loss in CG can be suppressed even further, as the survivors always lie within the lower range. This explains the fact that $K = 4$ and 5 MIN MFDs require the same dynamic range; any values beyond 100 can be safely saturated to 100 as they represent metrics of very unlikely paths in the trellis.

Table 3.1 Minimum required ACS dynamic range in MIN and MAX MFDs.

K	MAX ACS range in units	MIN ACS range in units
3	130	60
4	195	100
5	245	100

3.5.3 Effects of Errors in an Analogue MFD

In this section an investigation of the most common circuit-level errors arising from the use of analogue components in a MFD will be undertaken using system-level modelling. The errors of major concern in an analogue MFD mainly occur in the ACS block due to its sequential operation. These can be divided into comparator resolution and non-unity feedback-loop gain errors, which are commonly encountered in practice limiting the performance of a system. Other errors include SS non-unity gain errors which accumulate through each sample and hold stage in the analogue delay-line. In order to simulate the above errors independently and examine their individual effects on the decoder, a MIN MFD constructed using ideal Simulink® operations, for all blocks except the operation in study, will be considered in this section.

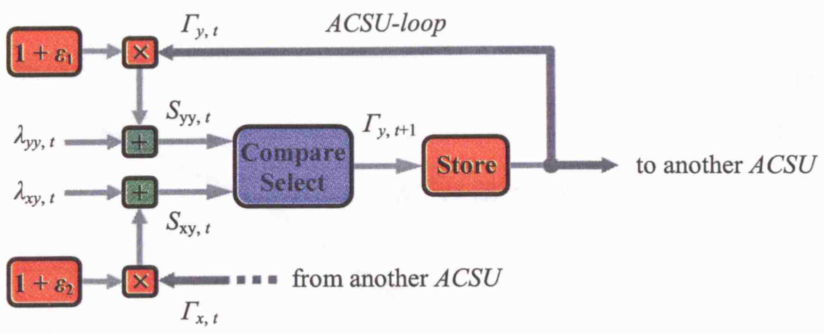


Fig. 3.14 ACSU model used for simulations of analogue implementation errors.

A MFD consists of 2^K identical ACSUs. Each ACSU is responsible for the comparison, selection and storage of the minimum of its two input values, each representing a different path metric (see section 3.2.2). In order to simulate the effects of the ACS errors mentioned above the model of fig. 3.14 is used, which shows an ACSU used in any $R = 1/2$ MFD.

3.5.3.1 ACSU Comparator Resolution Errors

The finite resolution of the comparator in an ACSU is modelled using a parameter *RS* in the *compare select* (CS) block of fig. 3.14, which represents the minimum detectable difference in its two inputs. The model works as follows: an ACSU selects the minimum of its inputs if and only if these differ from each other by at least *RS* units; otherwise it randomly selects a pre-fixed input as stated in equation 3.13. It is important to note here that the results obtained are not dependent upon the choice of the pre-fixed input (in this case $S_{yy,t}$) as no difference was observed when $S_{yy,t}$ and $S_{xy,t}$ were interchanged.

$$\Gamma_{y,t+1} = \begin{cases} min \ \{S_{yy,t}, S_{xy,t}\} & if \ \left|S_{yy,t} - S_{xy,t}\right| \geq RS \\ S_{yy,t} & if \ \left|S_{yy,t} - S_{xy,t}\right| < RS \end{cases} \qquad (3.13)$$

As previously discussed, the BMC output is normalised to a range of 0 to 8 units representing its finite saturation range. This means that the dynamic range (*DR*) of an ACSU cannot exceed $DR_{(ACSU)\max} = (2L) \cdot 8 = 16L$ units, as there are *L* branches of 2 bit metrics each in every decoding cycle. However table 3.1 shows that the required ACS range for a MIN MFD is less than $7L$. In any case it should be noted that this range is purely analogue as the CS input and output can take any value in between 0 and approximately $7L$. Simulation results indicate that the loss in CG in a $K = 3$ MFD

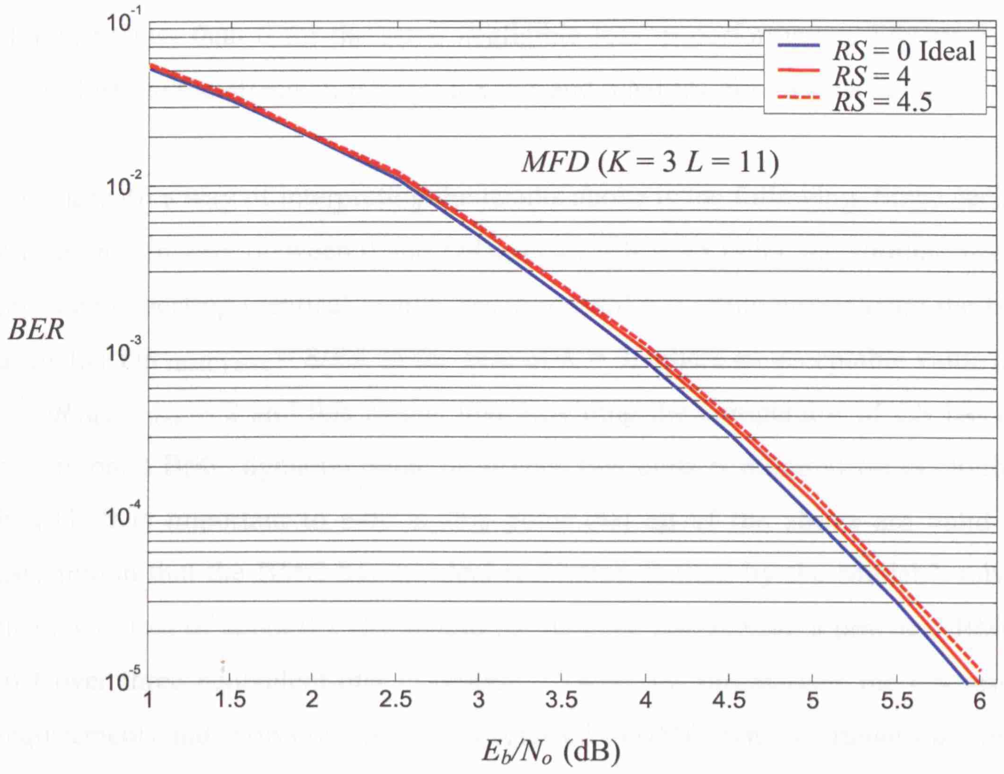


Fig. 3.15 Coding performance of the $K = 3$ MFD for various comparator resolutions.

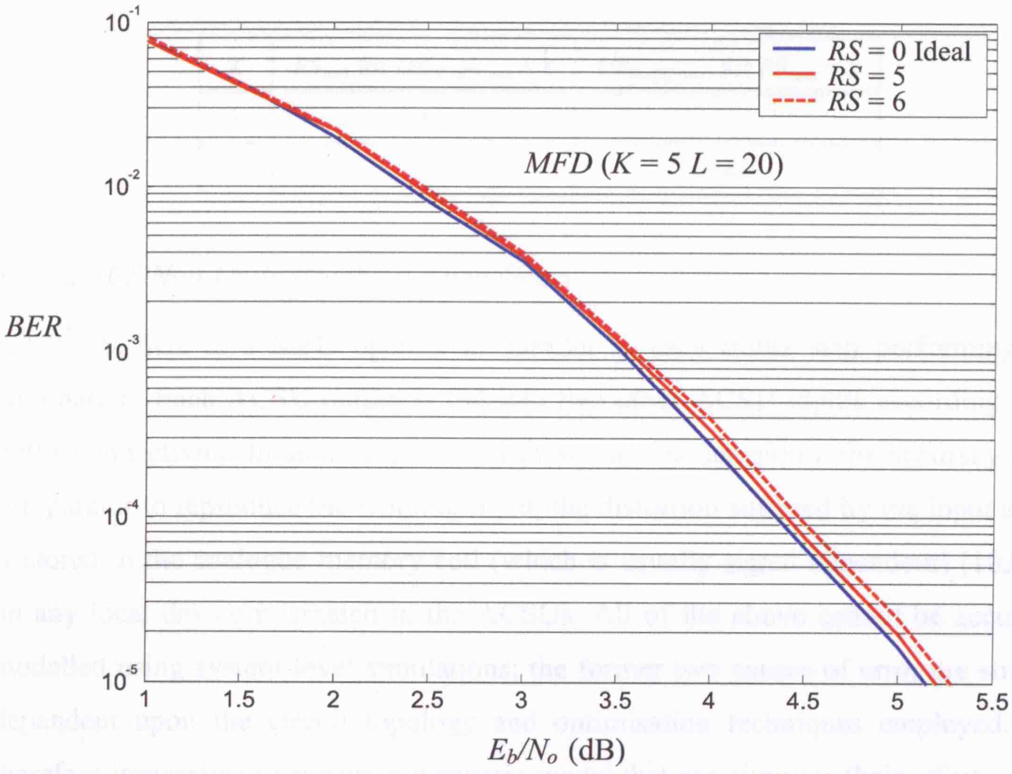


Fig. 3.16 Coding performance of the $K = 5$ MFD for various comparator resolutions.

is less than 0.1 dB at a BER of 10^{-5}, providing $RS \leq 4$. Also, for a $K = 5$ system, RS should be less than 6 for the same negligible loss in performance. Figures 3.15 and 3.16 illustrate the effects in BER of a $K = 3$ and 5 MFDs, for various values of RS.

An alternative way of interpreting the results above is the following. Since each BMC bit metric can vary between 0 and $DR_{(\mathrm{BMC})\ \max} = 8 > RS$ units, the simulations can be repeated expecting identical results, assuming $RS' = 1$ while normalising the BMC to a smaller $DR'_{(\mathrm{BMC})\ \max} = 8/RS$. In the case of $K = 3$, where an acceptable value of $RS = 4$, $DR'_{(\mathrm{BMC})\ \max} = 2$ and this means that providing the comparator of CS has a unity resolution, a BMC dynamic range of at least two units is required for negligible loss in CG. It is important to note at this point that all of the above are valid on the assumption that the BMC has an ideal resolution limited by the Matlab® solver and that a CG loss of about 0.25dB should be included considering a practical BMC with just over three equivalent-bits resolution. Table 3.2 summarises the CS resolution requirements and shows related results referred to BMC dynamic range for a practical MFD.

Table 3.2 CS resolution and BMC maximum range in a MIN MFD for less than 0.1dB loss in CG at 10^{-5} BER.

K	$RS_{\max}$ for $DR_{(\mathrm{BMC})\ \max} = 8$	$DR_{(\mathrm{BMC})\ \max}$ for $RS_{\max} = 1$
3	4	2
5	5.5	1.45

3.5.3.2 ACS Non-Unity Feedback-Loop Gains

All 2^K ACSUs in a MFD operate in parallel at each trellis step performing path elimination. Each ACSU output is fed into two other ACSU inputs according to the trellis connectivity. In an analogue ACS these outputs depend on the accuracy of the comparator to reproduce the winning input, the distortion suffered by the input after it is stored in the analogue memory cell (which is usually signal dependent) [103] and on any local device mismatch in the ACSUs. All of the above cannot be accurately modelled using system-level simulations; the former two causes of error are strongly dependent upon the circuit topology and optimisation techniques employed. It is therefore impossible to construct a generic model that can simulate their effects unless a certain design strategy is given from which circuit-level weaknesses can be identified and modelled [103]. The latter cause of error is a more general type that occurs in any VLSI circuit. Due to its nature, device mismatch is modelled statistically following the Gaussian distribution using circuit-level *Monte Carlo* (MC) simulations [104]. However, a general system-level statistical analysis combining all the above errors can be obtained, which can then determine circuit-level specifications (i.e. the maximum allowable deviation in each ACSU output). In order to investigate the statistical behaviour of the system in this manner and for simulation purposes, in this context mismatch is defined as the undesired variation in the ACSU outputs, such that each output has a non-unity Gaussian random gain factor of mean one. This is illustrated in fig. 3.14. The resulting metrics that enter the CS block in each ACSU can then be expressed as

$$\begin{aligned} S_{\mathrm{yy},t} &= (1+\varepsilon_1)\cdot\Gamma_{\mathrm{y},t} + \lambda_{\mathrm{yy},t} \\ S_{\mathrm{xy},t} &= (1+\varepsilon_2)\cdot\Gamma_{\mathrm{x},t} + \lambda_{\mathrm{xy},t} \end{aligned} \tag{3.14}$$

where ε_1 and ε_2 are Gaussian random variables of zero mean and σ standard deviation. The resulting analysis requires a large number of simulations for each σ as there are 2^{K+1} ε_i (i = 1, 2, 3, ..., 2^{K+1}) in a MFD generated once, at the beginning of each simulation, and several different ways in which they can be distributed in the ACSUs. In general and for reliable statistical precision, the BER results of at least five hundred simulations, for each σ, should be averaged up according to MC. This can be enormously time consuming, especially as K increases and hence not possible in practice. A more feasible approach was followed which assumes that ε_i change at every ACS cycle. In this case, only one simulation per σ is necessary and the sequential ε_i generation asymptotically has the effect of averaging the results of multiple simulations of fixed ε_i. The latter was verified by means of extensive simulation. Results are shown in the following pages illustrating the loss of CG at BER of 10^{-5} of the $K = 3$ and $K = 5$ MFDs, including simulation results which will be discussed next.

A more accurate but less realistic simulation of the effects of ACS feedback-loop gains is the worst case analysis approach. This is a theoretical evaluation of the decoder's performance under the worst case scenario that could possibly occur in the process of decoding a certain sequence. Since all possible transmitted codewords have

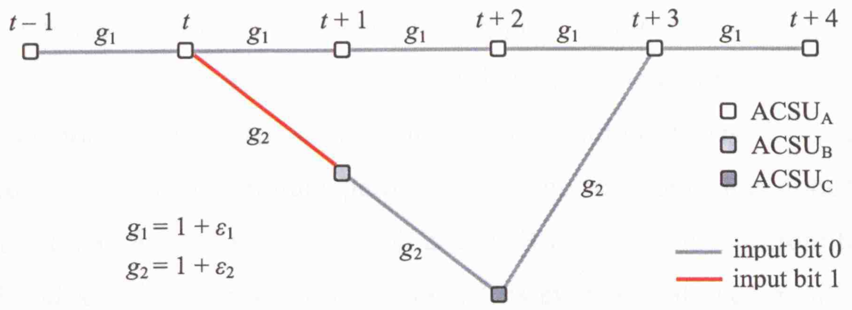


Fig. 3.17 Illustration of the ACSU non-unity feedback-loop gains on the all "0"s path and the shortest distance incorrect path for a $K = 3$ VD.

the same probability of error, the all "0"s sequence which results in the all "0"s codeword is conveniently selected to perform this study. In the case of the VD the ACS loop gain errors are modelled according to fig. 3.17. The loop gains g_1 and g_2 are introduced on the correct path, corresponding to the all "0"s sequence, and the shortest distance incorrect path of the trellis of fig. 2.9. The latter path is selected by the decoder only if $S_{AA,\ t+3} > S_{CA,\ t+3}$ (where S denotes the distorted path metrics according to fig. 3.14), resulting in one decoded bit in error. Therefore, the asymptotic probability of a decoding error can be found assuming the worst case where $\varepsilon_2 = -\varepsilon_1$ (ε_1, $\varepsilon_2 > 0$) [102]. However, in order to perform the same analysis in the MFD and since there is no storage of the history of the correct path a different approach is adopted. There are two related differences in this approach; first the shortest distance (to the all "0"s) incorrect path lies in the lower trellis and second the decoding decision is determined by the WTA block. This means that even when a different to the all "0"s path has the smallest metric at the end of a decoding cycle, providing it lies in the upper trellis the decoder will always output "0". Also in the MFD the first K steps are processed in parallel by the BMC where there are no ACS feedback-loop errors (see fig. 3.5). Despite that, there are some cumulative errors in the BMC introduced by the main feedback operation when one or more preceding output bits are decoded in error. Although these errors are not caused by any ACS imperfections, they indirectly limit its tolerance on the loop gains described above.

In order to investigate the effect of these errors in the performance of the MFD and establish BER upper bounds, the gains g_1 and g_2 of fig. 3.17 are placed on the all "0"s paths of the upper and lower trellises correspondingly. Two cases are studied, the worst case where $\varepsilon_2 = -\varepsilon_1$ and the case where $\varepsilon_2 = \varepsilon_1$ (ε_1, $\varepsilon_2 > 0$). The first case essentially favours the lower trellis since it multiplies its all "0"s path by $g_1 < 1$ and the correct path by $g_2 > 1$. It should be noted that since there are $(L - K)$ sequential ACS computations in every decoding cycle, the propagation of these loop gain errors is expected to severely deteriorate performance. The red solid and dotted lines in figures 3.18 and 3.19 show the loss in CG at a BER of 10^{-5} resulting from both cases in a $K = 3$ and $K = 5$ system correspondingly. It is evident that even a value of ε_1 as small as 0.5% has a non-negligible effect on the coding performance of the $K = 3$ MFD, which tremendously deteriorates as K grows, resulting in a loss of 1.86dB in a $K = 5$ system. This rapid degradation is a direct consequence of the increase of L, unlike in the VD where the shortest distance incorrect path increases with K. In fact, as K grows the sequential computations would decrease if L was not proportional to K in the MFD. This observation leads to the very simple solution of decreasing L especially when the advantage of a larger L cannot be exploited due to the imposed limitations from the use of analogue circuits. Figures 3.18 and 3.19 include results of the $L = 3 \cdot K$ MFD for both error cases. Although reducing the L is not always advantageous, as seen in the graphs, an optimum value of L can be determined for a specific ACS circuit as follows. Worst case ACSU output deviation results, using circuit-level corner analysis, determine the value of ε_1 from which extensive system-level simulations establish the optimum L.

ACS feedback-loop gain errors are probably the most severe limitation on the analogue implementation of convolutional decoders [102] and care should be taken on a design approach to ensure that these gain variations are kept to minimum. A more elaborate but very efficient solution to these errors in the MFD is the use of renormalisation as in the VD (see 3.2.2). Despite the need for the extra circuitry, employing renormalisation effectively translates to shortening the L or decreasing the required dynamic range in the system. Since the main cause of this rapid performance degradation is the path metric growth resulting in large dynamic range requirements in the ACS, the most efficient renormalisation scheme is the variable shift outlined in section 3.2.2. By subtracting the minimum state metric from all states at every ACS cycle the path metric expansion and hence the dynamic range is limited to the absolute minimum, which in turn limits the effect of the loop gain errors. The blue lines in figures 3.18 and 3.19 show the resulting loss in CG of the MFDs employing renormalisation, for the two error cases discussed above. As expected the loss is

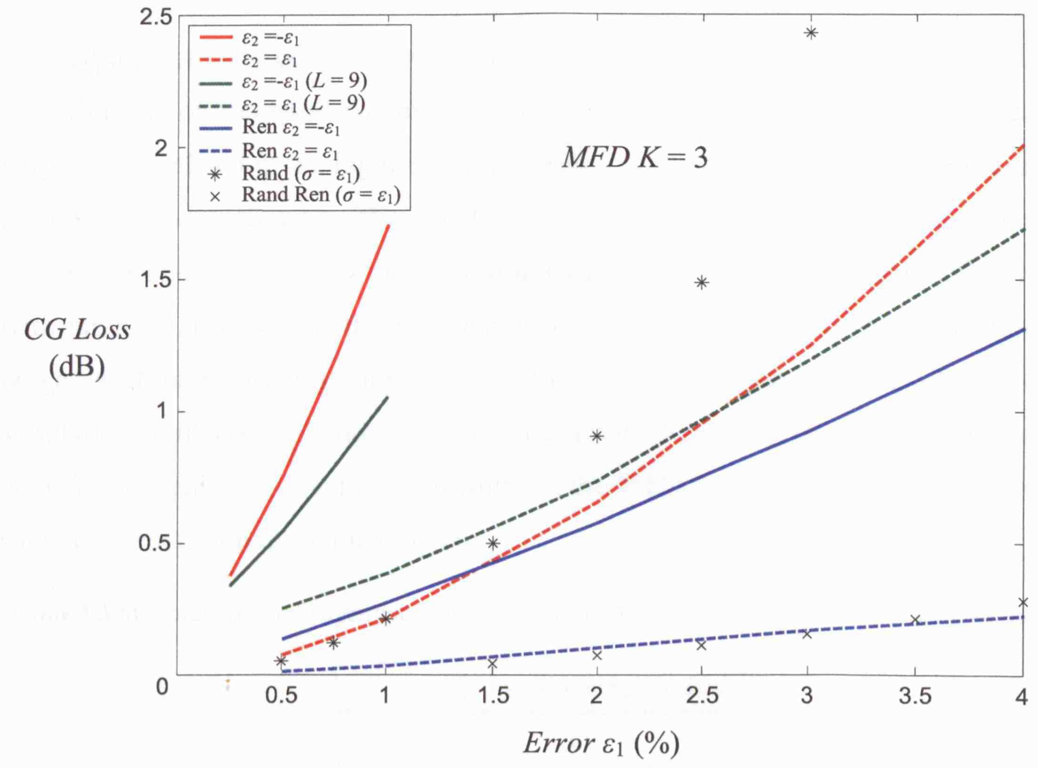


Fig. 3.18 Loss in CG in a $K = 3$ MFD including ACSU non-unity feedback-loop gain errors.

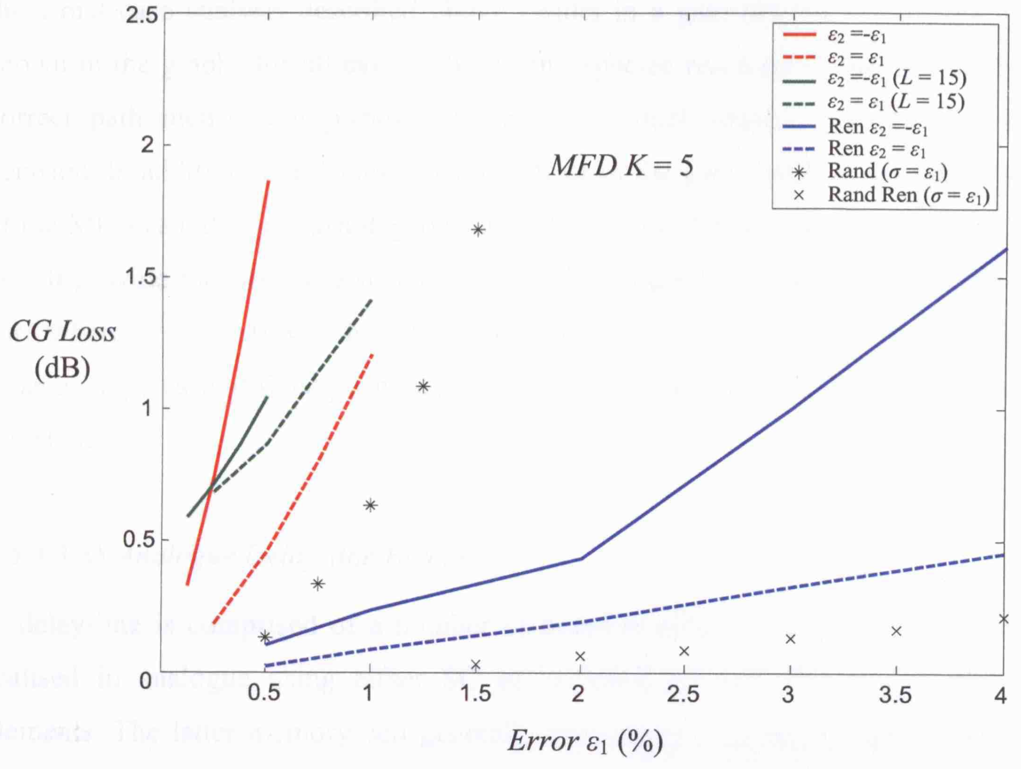


Fig. 3.19 Loss in CG in a $K = 5$ MFD including ACSU non-unity feedback-loop gain errors.

reduced dramatically compared with the loss of the MFDs without renormalisation at the extra cost of increased complexity. Nevertheless, the variable shift method can be very easily realised in analogue using an extra WTA circuit hence making the use of renormalisation even more efficient. Table 3.3 shows the dynamic range requirements of a $K = 3$ and $K = 5$ MFD incorporating renormalisation. Comparisons of table 3.3 with table 3.1 justify that the improvement in CG relies on the decreased dynamic range which in this case is not proportional to L. Finally, figures 3.18 and 3.19 include simulation results of the randomly generated errors discussed previously, the standard deviation of which is equal to ε_1. Results of the MFD employing renormalisation are shown as crosses on the same graphs.

Table 3.3 Minimum required dynamic range in MIN MFDs employing shift variable renormalisation.

K	MFD dynamic range in units
3	30
5	38

It is important to note as a final remark that reversing the sign of $\varepsilon_2 = -\varepsilon_1$ ($\varepsilon_1, \varepsilon_2 < 0$) in the worst case analysis described above results in a gain similar to the loss in CG shown in the graphs for all cases. This is an expected result as a consequence of the correct path metric multiplication by $g_1 < 1$, which clearly favours the correct decision. In addition, this implies that the effect of the gain errors on the performance of the MFD can also be advantageous and the fact that the results for $\varepsilon_2 = -\varepsilon_1$ with ε_1, $\varepsilon_2 < 0$ provide the best case scenario establishing upper BER bounds. Therefore the case where $\varepsilon_2 = \varepsilon_1 = 0$, which is the ideal case (i.e. no gain errors), can be thought of as an average case of fixed positive and negative gain errors ε_i distributed randomly in all ACSUs.

3.5.3.3 SS Analogue Delay-line Errors

A delay-line is comprised of a number of memory cells in cascade; these cells are realised in analogue using either SC or *switched current* (SI) sample and hold elements. The latter memory cell generally encounters large signal-dependent errors [103] and therefore, cascading L of these cells can result in a non-negligible effect on the coding performance of a MFD. Hence, the alternative SC sample and hold delay line is generally preferred. Simulation results verify that there is a minor, almost negligible, effect on the decoder coding performance even with the use of the most basic L sample and hold circuits in series. Non-unity values of $0.95 < g_{S/H} < 1.05$ representing the gain of each stage in an analogue delay line indicate that a $K = 3$ decoder is completely insensitive to this form of error. As L grows with K a small degradation in performance is observed, which is less than 0.25dB in a $K = 5$ MFD. This is anticipated as $L = 5 \cdot K = 20$ which means that the total distortion g_{total} of a received symbol, just before it is shifted out of the delay line that suffers a $g_{S/H}$ at each stage, is $g_{\text{total}} = (g_{S/H})^L$. Therefore, and in order for the SS errors to be negligible in a MFD, the design of a low power sample and hold cell of moderate speed that incorporates a $0.99 < g_{S/H} < 1.01$ is required.

CHAPTER 4

Analogue Decoding and Implementation of a MFD

The world we are living in is analogue (i.e. what we hear, see etc.); so is the transmission of signals in a digital communication system. A digitally encoded information sequence is represented as an analogue waveform (continuous in time and values) in order to be effectively transmitted. Including external interfering disturbances the received sequence looks rather like a random waveform, which is processed by dedicated analogue circuitry in the receiver front-end prior to analogue to digital conversion. Therefore two potential advantages of analogue decoding over digital arise already, the second being an apparent consequence of the first. Analogue decoders eliminate the need for large and power hungry *analogue to digital converters* in the front-end and as a result very high resolution can ideally be achieved. This in turn eliminates the quantisation noise encountered in analogue to digital conversion resulting in loss of CG. Also, analogue decoding can be very efficient especially when the employed algorithm requires simple arithmetic computations such as addition and multiplication [105]-[107]. Lower power dissipation and implementation complexity and sometimes higher operating speed can be more easily achieved with the use of simple analogue circuit techniques.

In this chapter, the potential advantages of analogue implementations followed by a brief overview of existing analogue decoders are outlined in the first two sections.

Section 4.3 is focused on the design of a CMOS mixed signal hard-decision MFD, intended as a proof of the MFDA principle. Circuit design issues, simulation results and experimental verification of fabricated chips are thoroughly covered. Measured performance characteristics of the MFD are also compared with existing analogue decoders. In the last section, circuit modifications and different structures of the SS and BMC circuits for the implementation of an all analogue soft-decision MFD are described.

4.1 Why Analogue?

Analogue decoding has been shown to be very efficient and closer to ideal decoding compared to digital methods since the late 1970's [102]. Nowadays, despite the advent and rapid evolution of advanced digital CMOS sub-micron integrated circuit (IC) technologies, analogue implementations of existing digital decoders have received much more attention in applications within and beyond convolutional codes [11], [10], [107]-[113]. Most of the early decoder examples were realised using BiCMOS circuits due to the *bipolar junction transistor* bandwidth and gain superiority in comparison with their CMOS equivalents; however, these decoders do not fully exploit the advantages of analogue implementation as further reduction in size and power consumption is possible in full CMOS designs. Also, BiCMOS processes have higher costs than conventional CMOS due to their complex fabrication and since CMOS transistors can be made smaller than bipolar, even further reduction in size and hence cost is achieved. Finally CMOS decoders are much more attractive from the point of view that they can be easily integrated with other CMOS components, such as digital circuitry, for single chip receiver solutions. Ideally these chips comprise the whole system (system-on-chip) including base-band, RF and digital signal processing.

As mentioned before the advantages of analogue decoding over digital lie mainly within the size (i.e. circuit complexity) and power and in some cases speed too. Analogue circuits of moderate resolution (up to six equivalent-bits) can be used more efficiently than digital counterparts and this is usually adequate in most decoders [8]-[9], [85]-[86] and [105]-[108]. In terms of power consumption, a digital 3-bit soft-decision decoder comprises the use, for example, of 6-bit adders and/or multipliers in order to accommodate the signal processing involved; this translates to six wires of regular rail-to-rail switching for every value representing a metric, which in analogue can simply be replaced by one. Therefore, and due to the low precision requirements, analogue decoder implementations are much better suited in terms of circuit complexity and die area too. A final remark on the speed of operation is that it has been demonstrated in several analogue decoders to be up to a few orders of magnitude higher than digital counterparts mainly due to the simplicity of the computations when performed in the analogue domain and the lack of ADCs [9]-[12], [108]-[109] and [111].

A common point of confusion in the literature is the mode of operation of analogue decoders, namely current and voltage mode. Although there is no fundamental difference between the two domains, there is always a more suited approach for a specific design structure that depends on system specifications, interface etc. In other words, although wherever there is current there is a voltage associated with it, the domain which the signal computations are performed in defines the mode of operation, which in turn depends on the decoder's architecture. Also, the mode of operation depends on the interpretation of the employed decoding algorithm. For example, simple arithmetic operations such as addition can be realised more easily in the current domain by joining branches together to a low-impedance node [11], whereas summing amplifiers or voltage to current converters are required in voltage-mode in order to perform the same operation [8]-[9] and [12]. One could argue that the simplest form of voltage to current converter is a resistor, however despite the feasibility of resistors in ICs, size and cost issues arise and in the case of standard CMOS processes matching and accuracy too. In addition replication of currents introduces errors due to transistor mismatches whereas a voltage can theoretically feed as many circuit nodes as necessary. In practice, in the latter case buffering is required to reduce charge transfers from one sub-circuit to another and to ensure impedance matching, while in the former case replication is achieved by current mirrors which at the same time serve the purpose of buffering. Therefore both modes of operation have their merits and limitations with the choice being reliant on the factors mentioned above.

4.2 Existing Analogue Decoders

4.2.1 Viterbi Decoders/Detectors

The most commonly employed decoding method in convolutional codes is the VD, as discussed in chapter 3. Applications beyond convolutional codes include *partial response maximum likelihood* (PRML) detection in digital magnetic recording (i.e. hard disk drive storage systems), where the VD performs maximum likelihood sequence estimation on a *partial response* (PR) channel [40]-[41] and [114]-[115]. Several analogue VDs have been reported in the literature [8]-[12] out of which the decoders in [9] and [11]-[12] successfully demonstrated all advantages discussed in the previous section. The term analogue VD refers to decoders that perform the BMC and ACS computations in the analogue domain (voltage or current mode); in practice, these decoders are mixed-signal as the SSM is always implemented in digital (see section 3.2) imposing size and power limitations [11]. In this section a brief comparison of recent IC design techniques in existing analogue VDs is outlined.

Table 4.1 summarises performance characteristics of various analogue VDs. The first analogue integrated VD was reported in [8] featuring high speed and low power CMOS circuits suitable for PRML detection in digital magnetic recording. The chip contains two independent dicode detectors that together operate on a class-IV PR channel. The BMC block consists of summing amplifiers using different *transconductance* (g_m) cells in order to calculate the path metrics. A differential input latched comparator is used to select the most likely path and update the digital path memory accordingly. Path metric normalisation is not required in this design as it employs the *difference metric algorithm*, which uses an alternative approach to perform the VA computations [116]. A more recent derivation of the difference metric algorithm was used in the VD of [9] for the same application. This design exploits fully the advantages of the difference metric algorithm and therefore eliminates the need for individual state metric computations. Two dicode Viterbi detectors are used in parallel, each operating at 100 Mb/s, resulting in a total decoding speed of 200Mb/s. The advantage of high speed and better matching properties of BJTs combined with the rail to rail capabilities of CMOS stages are attained with BiCMOS technology in this work. However, a major drawback here is the need for

Table 4.1 Performance characteristics of recent analogue IC VDs.

Ref.	Technology	Coding Scheme	# of states	Data rate (Mb/s)	Power (mW)	Energy (nJ/b)	Area (mm^2)
[8]	CMOS 2μm	Class-IV PR	2	50	89	1.78	3.24
[9]	BiCMOS 0.8μm	Class-IV PR	2	200	30	0.15	0.5
[10]*	CMOS 0.5μm	Convolutional	64	40	50	1.25	12.8
[11]	CMOS 0.8 μm	Convolutional	4	115	39	0.34	1
[12]	CMOS 0.25 μm	4-PAM PR	4	200	55	0.275	0.78

*Post-layout simulations

resistors used in the voltage to current converter and preamplifier stages, since they are usually difficult to integrate as discussed before. A more generic analogue VD intended for convolutional codes was presented in [10], where voltage-mode SC circuits were used for the BMC and the state-parallel ACS achieving a decoding speed of 50Mb/s according to post-layout simulation results. The first CMOS current-mode VD was reported in [11], using transconductors in the BMC effectively to convert the channel sampled data into current values proportional to branch metrics and a synchronous *replicating current comparator* (RCC) [105] together with two SI memory cells [103] for each ACSU. This is a much more attractive design approach for a general convolutional decoder as it features very high speed, low power and low complexity in a standard CMOS technology. Finally in [12] an analogue Viterbi detector was presented for multilevel *pulse amplitude modulation* (PAM) scheme. This type of modulation trades decoder complexity for channel efficiency (i.e. required bandwidth for a certain bit rate) and is commonly used in bandwidth limited systems [3]. Therefore, front-end quantisation is required whereas a combination of voltage and current-mode CMOS techniques utilize the decoder circuitry which occupies a die area of 0.78 mm^2.

It is important to mention here that the VDs intended for PRML employ a simplified version of the VA, hence direct comparison with a generic convolutional decoder is inequitable. Taking this into account, the work in [11] provides a solid justification for the analogue current-mode approach and the only limitation in performance is the digital SSM that accounts for more than 50% of the die area.

4.2.2 Turbo-Style Decoders

Turbo decoders employ algorithms which operate on graphs [107] and are generally known as *message passing algorithms*. These decoders have an *iterative* form as they are constructed from two or more constitute SISO decoders that exchange soft information with each other. A single bit (hard-decision) output is only produced after a certain number of iterations, increasing reliability. The SISO decoders usually realise the MAP decision rule employing the forward-backward algorithm also known as BCJR. However, maximum likelihood with SOVA can also be used which reduces the implementation complexity and has negligible loss in performance especially for low BER [74]-[75]. The most common form of BCJR implements the *sum-product* (SP) algorithm which performs additions of multiplications on bit probabilities in the backward and forward direction of a trellis [107]. This is best suited to analogue implementation as both computations involved can be achieved with as little as one transistor per computation using the translinear principle [111]. The SP algorithm can be realised with minimal complexity using the Gilbert multiplier [117] which exploits the exponential voltage-current relationship of BJTs or subthreshold CMOS transistors. Hence, Turbo style decoders employing the SP algorithm lend themselves to micro-power operation.

Table 4.2 shows a summary of performance characteristics of various existing Turbo style decoders in the literature. The pioneering work of [117]-[118] initiated the use of simple analogue circuit components in order to efficiently realise the SP algorithm. The first entirely analogue decoder based on the above was demonstrated in [119] using BiCMOS technology, verifying the merits over digital equivalent designs. The decoder implements a (18, 9, 5) tail-biting trellis code, which is a simpler form of code used in magnetic recording. The decoder of [120] is very similar to [119] realising a 2-state version of the code using a more advanced BiCMOS technology. The first functioning CMOS implementation of an analogue Turbo-like decoder was reported in [110]. Although the decoder was designed to operate in subthreshold, in practice, data rates beyond 1kb/s could not be attained. However, measured results in strong inversion revealed minor performance degradations. The decoders described so far, serve the purpose of proof of principle using the circuits of [117] which are based on the Gilbert multiplier. A different design approach for a larger code was presented

Table 4.2 Performance characteristics of existing analogue Turbo style decoders.

Ref.	Technology	Coding Scheme	Data rate (Mb/s)	Power (mW)	Energy (nJ/b)	Area (mm^2)
[119]	BiCMOS 0.8 μm	(18,9,5) Tail-Biting BCJR	100	50 (5V)	0.5	1.19
[120]	BiCMOS 0.25 μm	(16,8) Tail-Biting BCJR	320	20 (3.3V)	0.0625	0.12
[110]	CMOS 0.5 μm	(8,4) Tail-Biting BCJR	1	45.2 (3.3V)	45.2	0.81
[109]	CMOS 0.35 μm	R=1/3, N=48 Turbo	13.3	185 (3.3V)	13.9	1.32
[112]	CMOS 0.35 μm	R=1/3, N=132 UMTS Turbo	2	7.6 (3.3V)	3.8	4.07

in [109]. Very simple transconductors and CMOS maximum value circuits in weak inversion are adopted here that can work well even in moderate inversion incurring a small penalty of about 0.1dB. However none of the above decoders employ a powerful error correcting code adequate for a realistic application. A more elaborate scheme defined in the UMTS standard is employed in [112]. This is also a CMOS implementation consisting of two SISO decoders and realising the SP algorithm using the circuit techniques of [117], [119] and [109].

4.3 Design and Testing of a Hard-Decision MFD

There are potential benefits for the realisation of the MFD considering that in most existing analogue convolutional decoders, power and size reduction is limited mainly due to digital path memory requirements (see table 4.1). As a proof of principle for the functionality and feasibility of the new algorithm, the analogue computational core of a MFD, notably ACS and WTA blocks, needs to be designed for verification. To do this three mixed-signal hard-decision MFDs of $R = 1/2$ and $K = 3$ were designed and fabricated using the AMS 0.8μm and 0.6μm CMOS technologies and Cadence® suite. The main digital section in the decoders is the SS block and its

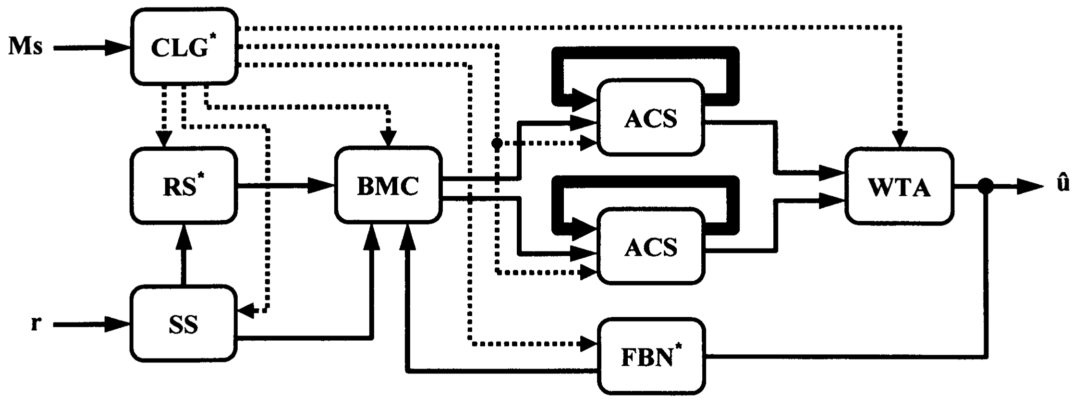


Fig. 4.1 Block diagram of a hard-decision MFD (*where CLG, RS and FBN are the clock generator, rotating switch, and feedback network blocks in this diagram and for illustration purposes only).

analogue realisation for a soft-decision MFD does not form a major constraint as it was seen in section 3.5.3 (see also section 4.4). Therefore, the same ACS and WTA circuits can be used for the construction of a soft-decision MFD by applying simple modifications in order to allow for higher dynamic range.

The functional block diagram of a hard-decision MFD is shown in fig. 4.1. This differs from the basic diagram shown in fig. 3.8 in the three following extra blocks: the *rotating switch*, *feedback network* and *clock generator*. The detailed operation of all these blocks including a *front-end* block (not shown in fig. 4.1) will be explained in the following subsections. Each subsection includes a brief high level description of a block, different circuit design approaches and implementation issues for the three fabricated chips. In section 4.3.9, the chips are assembled using the circuits of the previous sections, post-layout simulation results are shown and conclusions are drawn. In section 4.3.10, the laboratory test setup for verification of the chips will be described and measured results of BER against E_b/N_o and performance characteristics will be presented.

4.3.1 Front-End Circuitry

The input to the decoder is an analogue signal that results from the addition of white Gaussian noise on to the transmitted encoded data. The data is a bipolar signal waveform switching between $+a$ and $-a$ and can have any DC offset. Bipolar signalling is a much simpler form of BPSK modulation with identical BER performance over an AWGN channel. Hence, there is no need for a demodulator in the front-end.

The front-end circuitry consists of a sample and hold in order to synchronise the input corrupted by noise with the encoded data. The samples then need to be converted into binary values, since in the case of hard-decision there are only two levels of confidence (see section 2.1.3). This is achieved by the use of a *binary threshold detector*. The threshold needs to be set at the value of the bipolar signalling offset.

4.3.1.1 Front-End Circuit 1

Fig. 4.2 shows the circuit diagram of the front-end used in the first version of the chip. A simple sample and hold circuit, which consists of a capacitor and a switch (transmission gate) is used at the input, operating at the channel rate ($1/T_{S/H}$) which is twice the decoding rate ($1/T_D$) for $R = 1/2$. The sampled analogue values are then sent to the threshold detector which in this case is a clocked voltage comparator. The latter consists of a differential pair (M_5-M_6), used as a transconductance amplifier biased by I_{B1}, and a current steering CMOS latch (M_{11}-M_{14}). The amplifier converts the sampled voltage V_{IN} and the threshold voltage V_{TSH} into currents I_{IN} and I_{TSH}. Both currents are mirrored by M_8 and M_{10} respectively and then steered into the latch. When the voltage pulse F_{TD} is high the input/output node of the latch is at low impedance and the positive feedback is disabled. A competition is initiated between the currents at the falling edge of F_{TD}, when the latch goes into its regenerative mode. The difference of

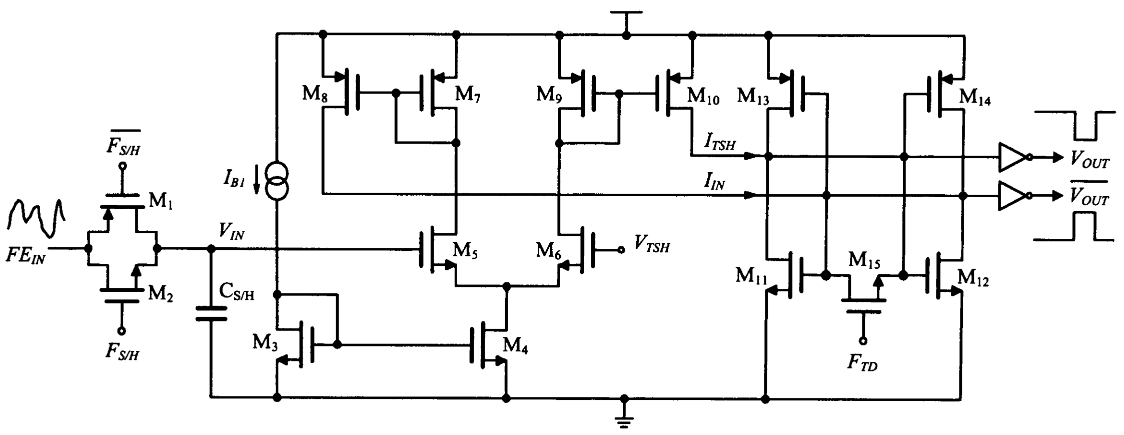


Fig. 4.2 Front-end circuit 1: sample and hold clocked voltage comparator.

Table 4.3 Transistor dimensions for front-end circuit 1.

Transistor	W (μm)	L (μm)
M_1-M_2	10	0.8
M_3-M_4, M_7-M_{10}	4	1.6
M_5-M_6	10	0.8
M_{11}-M_{12}	10	1.6
M_{13}-M_{14}	24	1.6
M_{15}	15	0.8

I_{IN} and I_{TSH} causes an initial voltage mismatch at the input/output nodes, which is rapidly amplified by the positive feedback resulting in a stable state output. When $V_{IN} > V_{REF}$, then $I_{IN} > I_{TSH}$ and V_{OUT} = "1"; otherwise V_{OUT} = "0". The two inverters, shown symbolically in fig. 4.2, are used to ensure similar track parasitic capacitance loading at the input/output nodes of the latch and also to give a low output when M_{15} is ON. Failure to ensure good matching of the total capacitances at the gates of M_{11}-M_{14} can lead to erroneous latch decisions as it will be further explained in section 4.3.5.

Table 4.3 shows the various transistor dimensions for the front-end circuit 1. The capacitor $C_{S/H}$ used for the sample and hold has a value of 0.5pF and the value of I_{B1} is 50μA. Also FE_{IN} is assumed to have an amplitude of 1V peak to peak and a DC offset of V_{TSH} = 1.5 V. As far as the timing operation is concerned, the pulse F_{TD} is a delayed version of $F_{S/H}$ sufficient for the amplifier PMOS mirrors to settle to the right values before the latch competition begins. Simulations show that the ON period of these pulses needs to be at least 5ns which is adequate time for $C_{S/H}$ to charge to FE_{IN} and for the gates of M_{11}-M_{14} to be brought to the same potential. The delay between the pulses can be as small as 2ns as this allows more time for the amplifier mirrors to settle.

Extensive Monte Carlo simulations show that the latch used in fig. 4.2 can resolve any $|I_{IN} - I_{TSH}| > 1.5\mu A$. Assuming M_5 and M_6 are perfectly matched devices in saturation, W and L are their channel widths and lengths correspondingly and that M_3 and M_4 form an ideal current mirror, then g_m is linearly related to $(I_{B1} \cdot W/L)^{1/2}$ [121]. The value of I_{B1} is set to 50μA in order to ensure that the differential transconductance g_m of the amplifier used here is high enough, such that $|I_{IN} - I_{TSH}| > 1\mu A$ for any differential input voltage $10mV \leq |V_{IN} - V_{TSH}| \leq 500mV$. The same value of g_m can be obtained with a smaller I_{B1}, and hence less power consumption, at the cost of increased *W*/*L* leading to a slower amplifier response and total front-end propagation delay.

4.3.1.2 Front-End Circuit 2

The circuit used in the second chip is shown in fig. 4.3. The same sample and hold as in circuit 1 is used in combination with a simpler clocked voltage comparator. Since a hard-decision decoder does not require a very high resolution comparator in the front-end, pre-amplification is unnecessary. Hence, the comparator of fig. 4.3 eliminates the transconductance amplifier of circuit 1 and the voltage inputs are applied directly on the gates of M_4 and M_5. Transistors M_6-M_9 form a latch very similar to the one used in circuit 1. The difference here is that the switch M_{15} (see fig. 4.2) is replaced with switches M_{10} and M_{11}. The latter has the advantage of using one less clock signal resulting in a less complex clocking scheme (see 4.3.8). In the reset phase, when $F_{S/H}$ is high, V_{IN} is applied to M_4 while charging $C_{S/H}$ and M_{10}-M_{11} pull the gates and drains of M_6-M_9 to the positive supply. At this stage M_3 is off so there is no current flowing through the comparator. When $F_{S/H}$ goes high, M_3 is ON sourcing current to M_4-M_5 in order to compare V_{IN} and V_{TSH}. The current mismatch in the drains of M_4-M_5 (I_{DM4} and I_{DM5}) causes the latch to switch to a stable state such that if $V_{IN} > V_{TSH}$ then $I_{DM4} > I_{DM5}$ and V_{OUT} is high, otherwise V_{OUT} is low. After switching to a stable state there is no static current flowing in the circuit which makes it very low power.

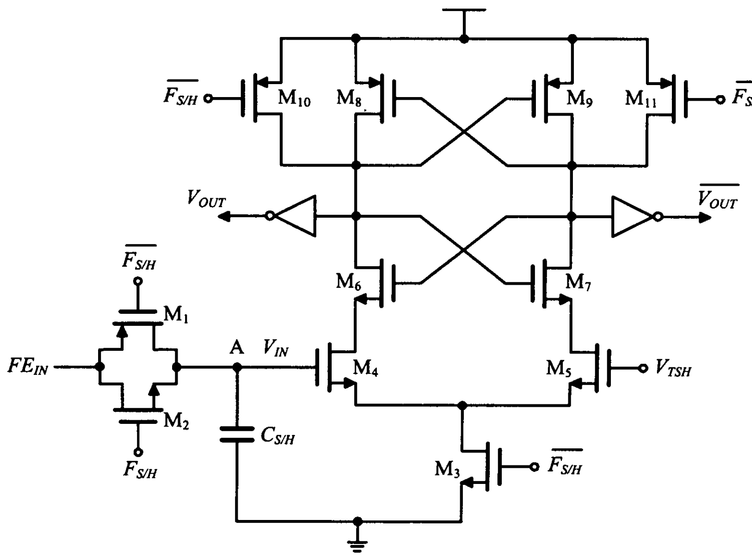


Fig. 4.3 Front-end circuit 2: sample and hold clocked voltage comparator.

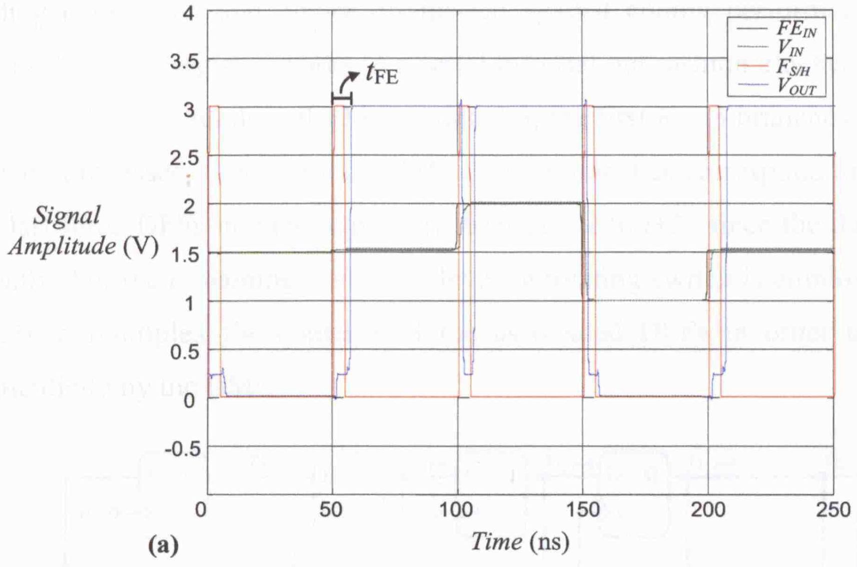
Table 4.4 Transistor dimensions for front-end circuit 2.

Transistor	W (μm)	L (μm)
M_1	30	0.6
M_2	15	0.6
M_3, M_{10}-M_{11}	6.4	0.6
M_4-M_5	12	1.5
M_6-M_7	4	1.2
M_8-M_9	7	1.2

Table 4.4 shows the transistor dimensions for circuit 2. It is important to note here that M_1 is twice as large as M_2 in order to compensate for the *clock feed-through* (see section 4.3.5.2) induced on node A due to capacitive coupling from M_3. This is explained as follows. When $F_{S/H}$ is high, M_4-M_5 are OFF and $C_{S/H}$ is charging to FE_{IN}. When $F_{S/H}$ goes low, M_3-M_4 are ON and the associated parasitic capacitances at node A now include the gate to source and gate to drain capacitances of M_4 and M_3 correspondingly. The latter capacitances discharge node A resulting in a $V_{IN} < FE_{IN}$. Using a larger M_1 ensures that the extra charge for the parasitic capacitances in the regenerative mode is provided from the gate to drain capacitance of M_1. The rest of the specifications including V_{IN}, V_{TSH} and $F_{S/H}$ are as in circuit 1.

Circuit 2 uses a dynamic latch with zero static power dissipation as there is current flowing only during the regeneration phase. In contrast, circuit 1 uses a static latched comparator driven by a pre-amplifier, which needs a constant bias current I_{b1}. Since the latch is driven by PMOS current sources, after regeneration the smaller of the two input currents flows through one of M_{11}-M_{12}, increasing static power even more. However, the *kickback noise* is more severe in circuit 2 as the drains of the differential pair (M_4-M_5) are connected directly to the latch providing a capacitive path from V_{IN} to the source of M_6, where the voltage variation is large [122]. Kickback noise generally results in a limited comparator resolution. System level simulations show that a resolution of 20mV in the front-end of a hard-decision MFD, which is about 4.6 equivalent-bits for an input range amplitude of 500mV, results in no *CG* loss. This resolution can be easily achieved using circuit 2 including clock feed-through and kickback errors according to Monte Carlo simulations.

Figure 4.4 shows the transient response of the complete front-end circuit 1 and 2 simulated using SPECTRE models provided by AMS CMOS 0.8µm and 0.6µm technologies respectively. The power supply is 3V and the circuits are clocked at 20MHz. The average power dissipation of circuit 1 is about 400µW whereas that of circuit 2 is 55µW. The maximum propagation delay t_{FE} including the $F_{S/H}$ width, as shown on the graphs in fig. 4.4, is 8.2ns for circuit 1 and 8ns for circuit 2. The sample and hold error in circuit 1 is less than ± 5mV and is mainly due to clock feed-through induced by M_1-M_2. This leads to a total of 10mV resolution for circuit 1 which is half the simulated resolution of circuit 2.



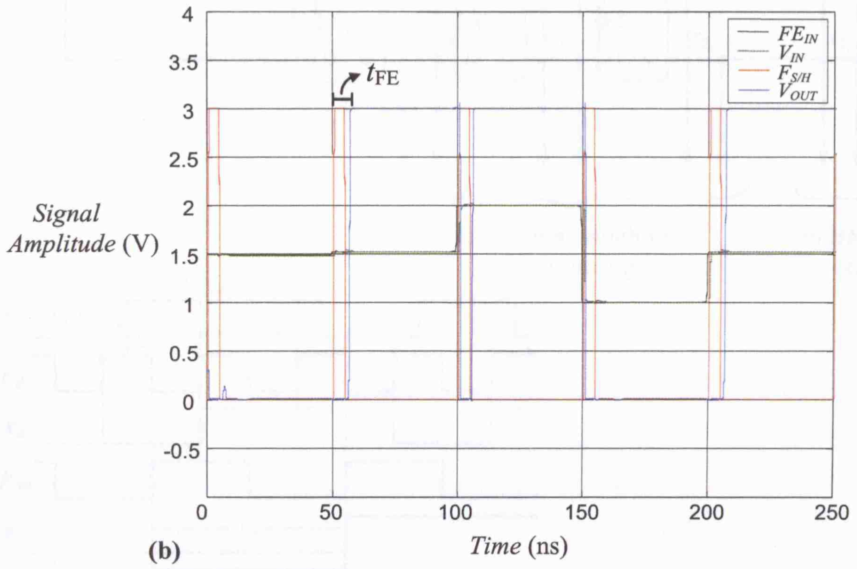


Fig. 4.4 Transient response of front-end (a) circuit 1, (b) circuit 2

4.3.2 SS Circuit

The 1-bit quantised outputs of the comparator of the front-end circuitry form the received sequence of code bits **r** (see fig. 2.1). This sequence needs to be demultiplexed in order to discriminate $\mathbf{r}_1$ from $\mathbf{r}_2$ and store each vector on a separate delay line. A standard *delay flip-flop* (DFF) is used to initially delay the received sequence by ¾ $T_{S/H}$. The delayed sequence $\mathbf{r}_d$ and **r** are then sampled at the same time and stored in two shift registers consisting of DFFs. Fig. 4.5 shows the complete SS block and timing diagrams. The upper register stores a block of $\mathbf{r}_1$ and the lower register stores a block of $\mathbf{r}_2$. The block length of each register is $L = 10$ effectively trading complexity and power dissipation against coding performance (see section 3.5.1). The shift register works in a serial in/serial out fashion clocked at the data rate $1/T_D = 1/(2 \cdot T_{S/H})$. As described in section 3.3, the first $K = 3$ branches of the modified tree are processed directly by the BMC. These branches correspond to the contents of the last three DFFs in each register (subscripted t to t+2) since the data is coming in serially. For the remaining $L - K = 7$ levels a rotating switch is employed (see section 4.3.3) to multiplex the contents of the associated DFFs in order to be processed sequentially by the BMC.

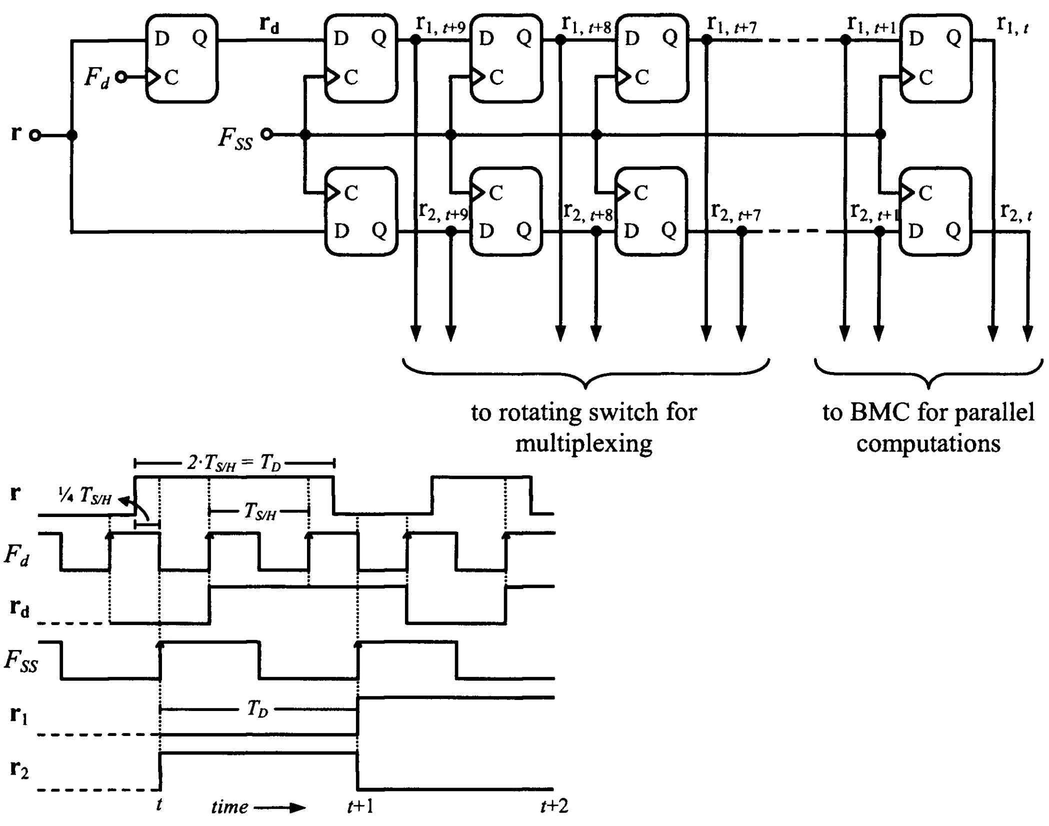


Fig. 4.5 The symbol storage circuit and timing diagrams.

The total propagation delay imposed by the SS block t_{SS}, is 62.5% of the decoding cycle T_D as illustrated in the timing diagram of fig. 4.5. Once a new bit is shifted into each register, the oldest one is shifted out. Therefore, the SS needs to be clocked L = 10 times in order for all the received code bits (associated with the first data bit) to be shifted in. Hence, the decoder has an initial delay of

$$T_{total} = t_{FE} + t_{SS} + 10 \cdot T_D \qquad (4.1)$$

before it starts to output relevant information. T_D is the time it takes for the contents of the SS to be processed by the following blocks of the MFD and it therefore defines the decoding speed. T_D is mainly determined by the maximum operating speed of the BMC and ACS blocks as it will be seen in the following sections.

4.3.3 Rotating Switch Circuit

In order to implement the necessary multiplexing for the contents of the first seven stages of the SS, the rotating switch of fig. 4.6 is used. As illustrated, there are fourteen different switches controlled by seven clock pulses. In this way, two received code bits (one from each register) are selected at each clock pulse. The switches are implemented using standard transmission gates of minimum geometry. The input

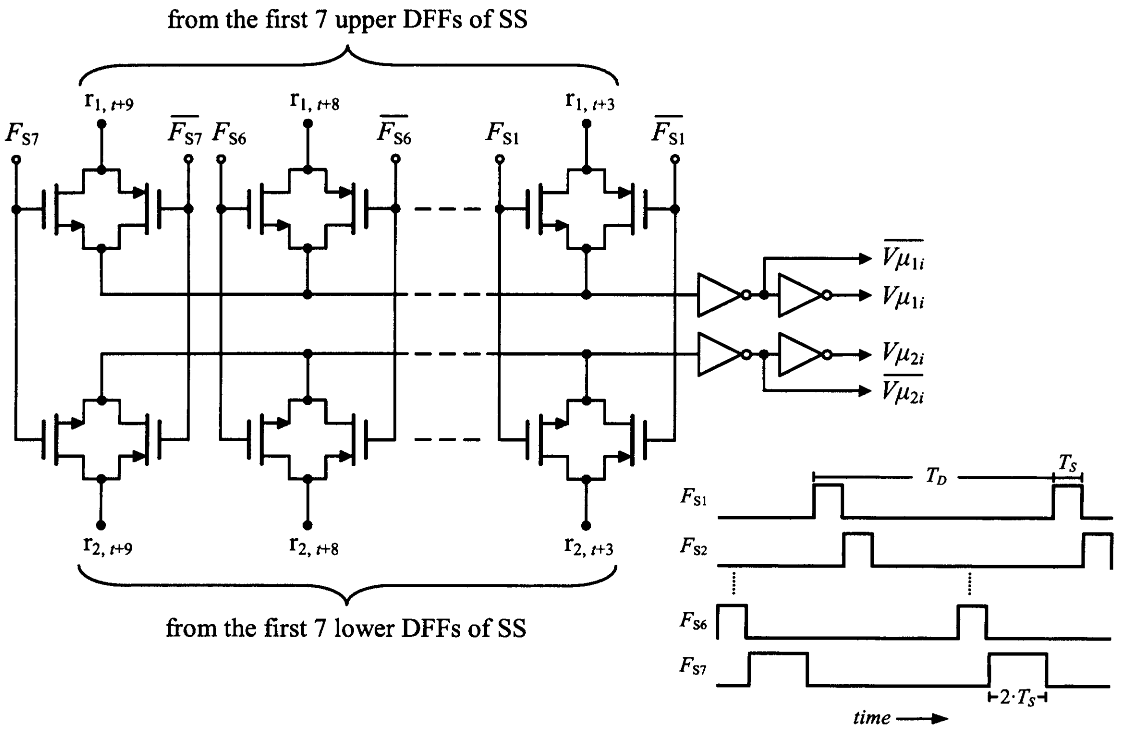


Fig. 4.6 The rotating switch circuit and timing diagrams.

sides of the seven upper (or lower) switches connect to the taps of the upper (or lower) register of fig. 4.5, while the output sides are connected together. The resulting output $V_{\mu 1i}$, $V_{\mu 2i}$ and their compliments form the control voltages for the sequential processing in the BMC, where $i = 3, 4, 5\ldots, 9$.

The clocks F_{S1}-F_{S6} have a duty cycle of $T_S/T_D = 12.5\%$ whereas the duty cycle for F_{S7} is 25%. The duration of the former pulses T_S is determined solely by the total time the BMC and ACS require for an output, whereas pulse F_{S7} needs to include extra time for the WTA. Although, an extra T_S may not be required (depending on the WTA architecture), it is convenient to generate a pulse with twice the duty cycle of F_{S1}-F_{S6}. This translates to a trade off between speed of decoding operation and clock generator complexity (see section 4.3.8).

4.3.4 BMC Block

The choice of the BMC analogue processing domain, where the signals are manipulated in, depends on the mode of operation of the ACS (current or voltage mode). As described in previous sections, in a hard-decision MFD, stored in the SS are the received code bits. However, this decoder serves as a proof of principle for the all analogue soft-decision MFD, in which the SS stores analogue samples. Taking this into account, an optimum design strategy for the analogue core (ACS and WTA), which is the key building block, is adapted accordingly. The current-mode operation is the preferred approach for realising the analogue core and hence the BMC, as it will be further explained in section 4.3.5.

The BMC architecture also depends on the decoder configuration. Hamming distance is employed by a MAX MFD and inverse Hamming distance is employed by a MIN MFD as it was described in sections 3.2.1 and 3.5.2. In the first implementation of the MFD, a MAX configuration is adopted since it is determined by the operation of the analogue core (select maximum circuits were used in ACS and WTA). In the following two versions of the chip, the analogue core is modified to perform select minimum operations exploiting the power efficiency discussed in section 3.5.2.

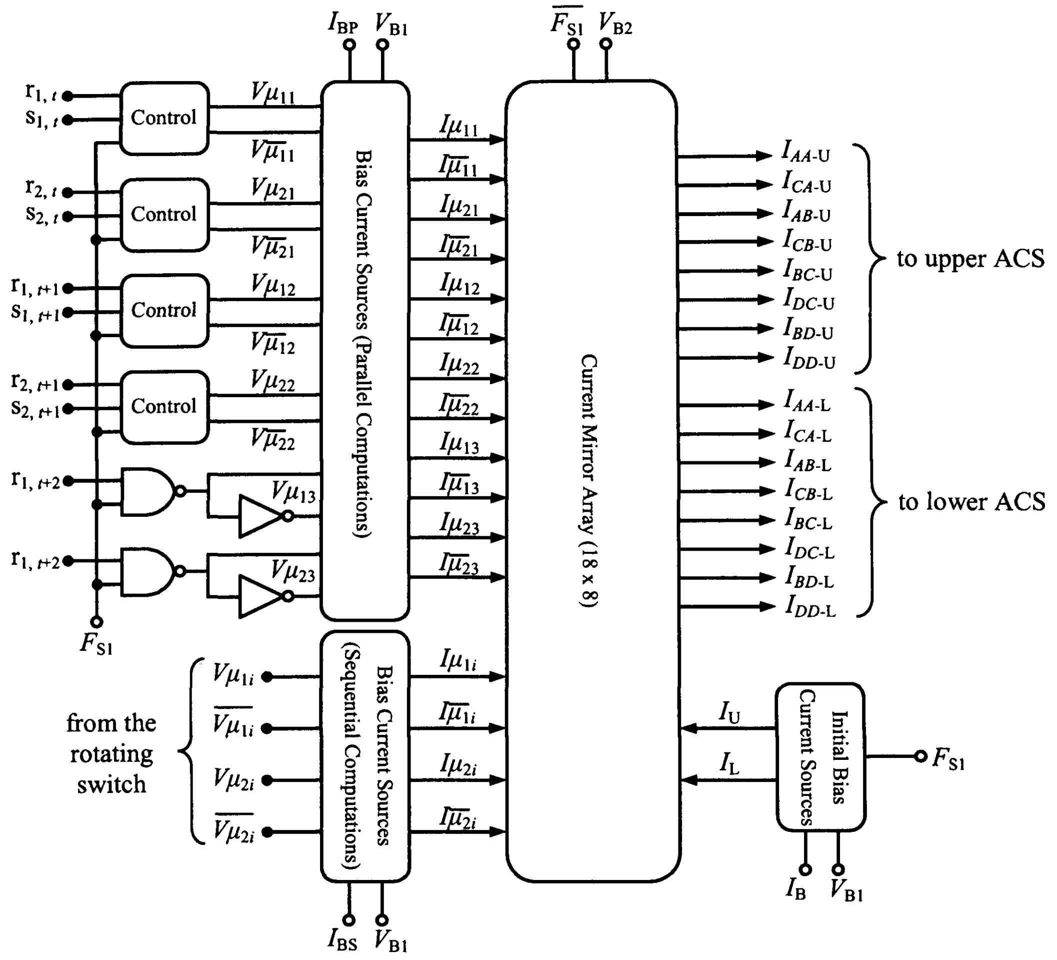
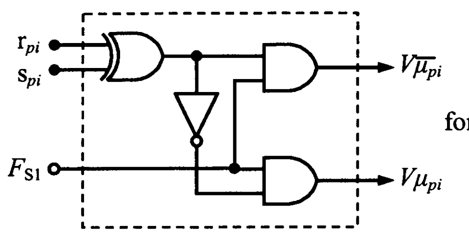

I_{BP} V_{B1}
$\overline{F_{S1}}$ V_{B2}
Control
Bias Current Sources (Parallel Computations)
Current Mirror Array (18 x 8)
to upper ACS
to lower ACS
F_{S1}
from the rotating switch
Bias Current Sources (Sequential Computations)
Initial Bias Current Sources
I_U
I_L
I_B V_{B1}
I_{BS} V_{B1}

Fig. 4.7 The detailed BMC block diagram.

The BMC block is divided into three main sub-blocks: the control, the biasing and the metric generator circuitry. The control circuit is used for the parallel computations where the corresponding branch metrics are variable (see fig. 3.5). It uses the outputs of the last two levels of the SS ($\mathbf{r}_t$ and $\mathbf{r}_{t+1}$ in fig. 4.5) and information from the feedback network in order to calculate the programmable bit metrics in digital domain. The metric generator consists of an array of current mirrors with multiple outputs interconnected in a manner that resembles all branches of the modified tree. The output pulses of the rotating switch and control circuits are used to switch on two bias sources supplying current to the mirrors of the metric generator only when necessary. These bias sources including another source, which is used only at the beginning of each decoding cycle, comprise the biasing circuit. Fig. 4.7 illustrates a complete diagram for the BMC. The design details for all the sub-blocks will be discussed next.



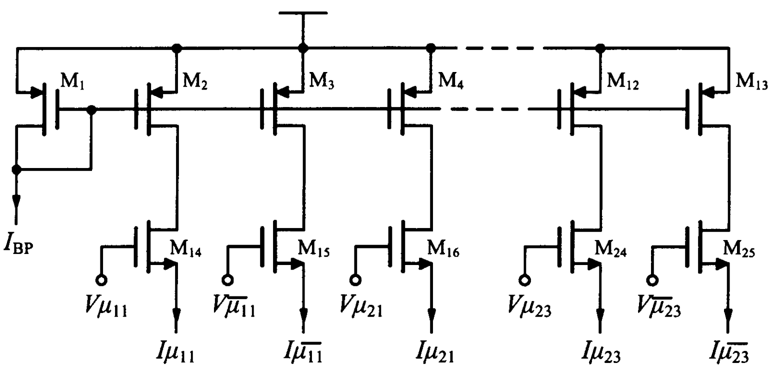
for p, $i = 1, 2$

$V\mu_{pi}$	$\overline{r_{pi}\oplus s_{pi}}$	r_{pi}	s_{pi}	F_{S1}
0	1	0	0	0
1	1	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	1	1	1	0
1	1	1	1	1

Fig. 4.8 The control circuit and truth table for the realisation of Hamming Distance.

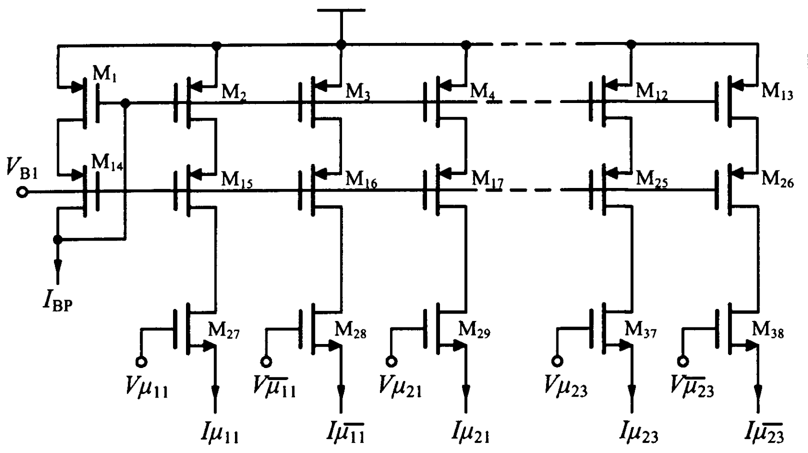
The control circuit associated with a single bit metric for the first two levels in the modified tree consists of three standard logic gates and an inverter as shown in fig. 4.8. r_{pi} is the received code bit stored in the i^{th} (from the last) DFF of the p^{th} register of the SS and s_{pi} is its associated sent code bit reproduced in the feedback network. An XNOR function between the sent and received code bits is used to realise the Hamming distance shown in the table of fig. 4.8 (see also equation 3.9). This function and its compliment need to be activated only in the first cycle of the decoding operation, in order to generate the programmable branch metrics (see fig. 3.5). This is achieved using the AND gates and the clock pulse F_{S1}. Note that the control circuit of fig. 4.8 is used in a MAX MFD. For a MIN MFD, the outputs of the circuit need to be interchanged, since the inverse Hamming distance is realised using an XOR.

Fig. 4.9 shows the schematic diagram of the bias current source used for the parallel computations in the first version of the chip. A simple PMOS current mirror with twelve outputs and single NMOS switches are used. The bias current I_{BP} is generated externally and is set to 10μA. The switches are controlled by the generated voltage metrics as depicted in fig. 4.7, effectively converting them into current metrics.



Transistor	W (μm)	L (μm)
M_1-M_{13}	4	1.6
M_{14}-M_{25}	2	0.8

Fig. 4.9 Realisation of the first version of bias current source (parallel computations) of fig. 4.7.



Transistor	W (μm)	L (μm)
Version 2		
M_1-M_{13}	4	1.2
M_{14}-M_{26}	4	0.8
M_{27}-M_{38}	0.8	0.6
Version 3		
M_1-M_{13}	4	3.6
M_{14}-M_{26}	8	0.6
M_{27}-M_{38}	1	0.6

Fig. 4.10 An improved version of fig. 4.9 used in the second and third chips.

The rest of the bias current sources shown in fig. 4.7 are identical to the one shown in fig. 4.9 with less output branches.

An improved version of the mirrors is used in the bias current sources in the second and third versions of the chip, providing higher output impedance and hence better accuracy. Fig. 4.10 shows the current source for the parallel computations employing wide swing cascode mirrors. The bias voltage V_{B1} is 1V in both versions of the circuit whereas I_{BP} is 8μA and $4\mu A \leq I_{BP} \leq 8\mu A$ in versions 2 and 3 respectively. From the transistor dimensions shown in the table of fig. 4.10 it can be seen that version 3 uses a much wider gate length for the mirror transistors to improve matching of the current metrics. Also, the aspect ratio (W/L) of M_1 is three times smaller in version 3 in order to increase its gate to source voltage $V_{GS(M1)}$ and ensure M_1 is in strong inversion.

The current mirror array shown in fig. 4.7 consists of eighteen independent current mirrors of multiple replication outputs each. A single output of nine mirrors is used for each ACSU input, six for the parallel metrics associated with the first three branches, two for the remaining sequential metrics of a decoding cycle and one mirror output for the initial bias current supply. There are eight path metrics just before the first path elimination in each sub-tree (see fig. 3.7). For a MAX hard-decision MFD implementation, the bit metric at any level of the modified tree is given by

$$\mu_{pi} = \begin{cases} I\mu_{pi}, & \text{if } s_{pi} = 1 \\ I\bar{\mu}_{pi}, & \text{if } s_{pi} = 0 \end{cases} \tag{4.2}$$

A closer study of fig. 3.5 leads to the observation that for the generation of all sixteen path metrics for the upper and lower sub-trees, eight $I\mu_{pi}$ and eight $I\bar{\mu}_{pi}$ are required for each combination of p = 1, 2 and i = 1, 2,...,4. This means that each mirror needs to replicate its output eight times.

An NMOS version of the high swing cascode mirror of fig. 4.10 is used for all the mirrors in the BMC array of all three chips. This is shown in fig. 4.11. There is an amplification factor α between input and output currents in the mirror which is less than unity depending on the version of the chip (i.e. bias current I_{BP} and I_{BS}) and the required analogue core resolution. All mirrors have the same amplification factor apart from the two initial bias mirrors which have the same aspect ratio resulting in α = 1. In the twelve mirrors responsible for the parallel computations there is a switch driven by the compliment of F_{S1} to ensure they do not interfere with the subsequent sequential computations. A single NMOS switch is used to discharge the common gates of M_1-M_8 in each mirror.

Parameter	Version 1	Version 2	Version 3
V_{B2} (V)	2	1.75	1.5
$I\mu_{pi}$ (μA)	10	8	4-8
α	0.1	0.25	0.25

	Version 1		Version 2		Version 3	
Transistor	W (μm)	L (μm)	W (μm)	L (μm)	W (μm)	L (μm)
M_1	20	1.6	16	1.2	16	3.6
M_2-M_8	2	1.6	4	1.2	4	3.6
M_9-M_{16}	4	1	2	0.8	4	0.6

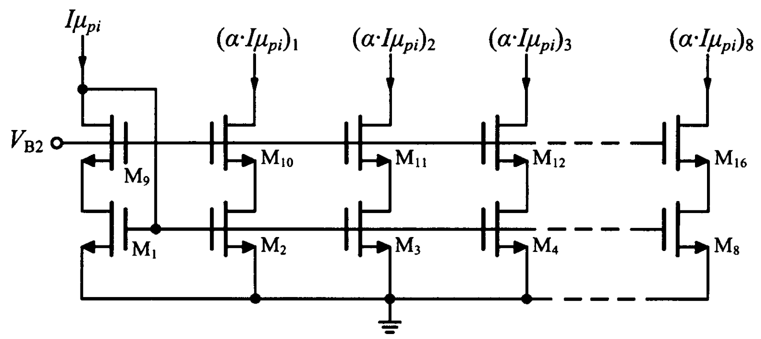

$I\mu_{pi}$
$(\alpha \cdot I\mu_{pi})_1$
$(\alpha \cdot I\mu_{pi})_2$
$(\alpha \cdot I\mu_{pi})_3$
$(\alpha \cdot I\mu_{pi})_8$
V_{B2}
M_9
M_{10}
M_{11}
M_{12}
M_{16}
M_1
M_2
M_3
M_4
M_8

Fig. 4.11 A cell of the current mirror array of fig. 4.7 and parameter tables for all three versions.

In summary, each of the sixteen outputs of the BMC shown in fig. 4.7 is the result of the addition of nine currents coming from different mirrors of the type shown in fig. 4.11. This addition is in fact performed in the ACSUs accordingly. For example, $I_{AB\text{-L}}$ and $I_{CB\text{-L}}$ are the two input currents in the lower ACSU$_{\text{B}}$ which carry the path metrics in the first cycle (F_{S1}) or the branch metrics at any other cycle (F_{S2}-F_{S7}) for the trellis transition from state A to B and C to B correspondingly. The total propagation delay of the BMC, T_{BMC}, is the necessary time for the generation of a bit metric $\alpha \cdot I\mu_{pi}$. The latter is fixed to 1μA and 2μA in the first and second versions of the chip correspondingly, according to the table of fig. 4.11. In version 3, it is made variable in order to externally control the resolution of the analogue core. Simulation results show that in the first two versions of the BMC, T_{BMC} is about 18ns, whereas in the third version T_{BMC} is 25ns and 42.5ns for $I\mu_{pi}$ 8μA and 4μA respectively. T_{BMC} is very critical as it partly determines the duty cycle T_{S} (see fig. 4.6). Therefore, version 3 trades operating speed for matching accuracy.

4.3.5 ACS Block

The ACS block is the most critical circuit in the system as it continuously performs sequential computations based on previous results. Accumulated errors resulting from the feedback-loop can degrade overall performance significantly to the point where the use of the decoder is redundant (see section 3.5). Therefore ACS precision and speed of operation are very important in determining the characteristics of a MFD. The ACS block consists of eight identical ACSUs very similar to the ones shown in fig. 3.3. One unit is used in each state of the modified tree. An ACSU consists of two summers, a 2-input *min* or *max* operator and two memory cells. An analogue *min/max* operator is defined here as the circuit whose output is the minimum or maximum of its inputs and it is necessary to execute the CS operation. Voltage or current-mode circuits can be used for the realisation of all or some of the above sub-blocks, as discussed in section 4.1. If the addition is performed in the current domain there is no need for any summers providing the ACSU feedback-loop is in the same domain. This leads directly to a preliminary conclusion that the complete current-mode approach has a major advantage over a voltage-mode implementation. However, a brief overview of potential voltage-mode circuits will be discussed next to further support this statement.

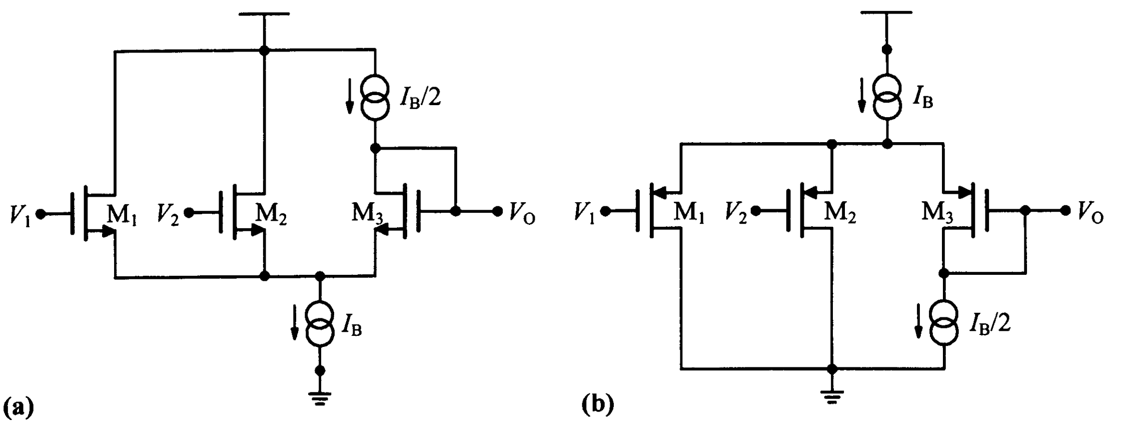


Fig. 4.12 CMOS voltage-mode **(a)** max and **(b)** min circuits based on the common-emitter configuration of [124].

A common implementation of a voltage-mode max circuit is based on the common source configuration of fig. 4.12(a). In this circuit the maximum of V_1 and V_2 appears at the common source shifted down by a V_{GS} drop. Only one of M_1-M_2 will be on, forming a unity gain feedback differential pair with M_3. Hence, $V_O = \max\{V_1, V_2\}$. Similarly, $V_O = \min\{V_1, V_2\}$ in the min circuit of 4.12(b). Both circuits can be extended to include multiple inputs for use as WTA operators. However, they suffer from large *corner errors* and usually require compensation to improve frequency response [123]. Corner error is the phenomenon whereby when the difference of the inputs is very small, failure to reproduce the winning input occurs and the output is usually near the average of the two inputs. The configuration of fig. 4.12 originates from the popular common emitter principle of [124]. The implementation of the exact circuit using bipolar transistors provides higher precision and speed due to their bandwidth and gain superiority over CMOS. A CS circuit using the same principle was adopted in [125]. In order to improve the circuit characteristics in CMOS designs several modifications and different techniques have been proposed in the literature resulting in complex structures (involving multiple voltage-to-current conversions) and/or increased power consumption [123] and [126]-[128].

On the other hand, traditional max circuits featuring current inputs are based on WTA circuits, such as the Lazzaro circuit [100], modified to replicate the min/max current at the output. An example of this type of circuit is shown in fig. 4.13(a). The operation of this circuit is based on the principle of a MOS transistor in saturation conducting the maximum drain current for a certain gate to source voltage. Therefore, the shared gate voltage of M_1 and M_3 corresponds to the saturation value imposed by the

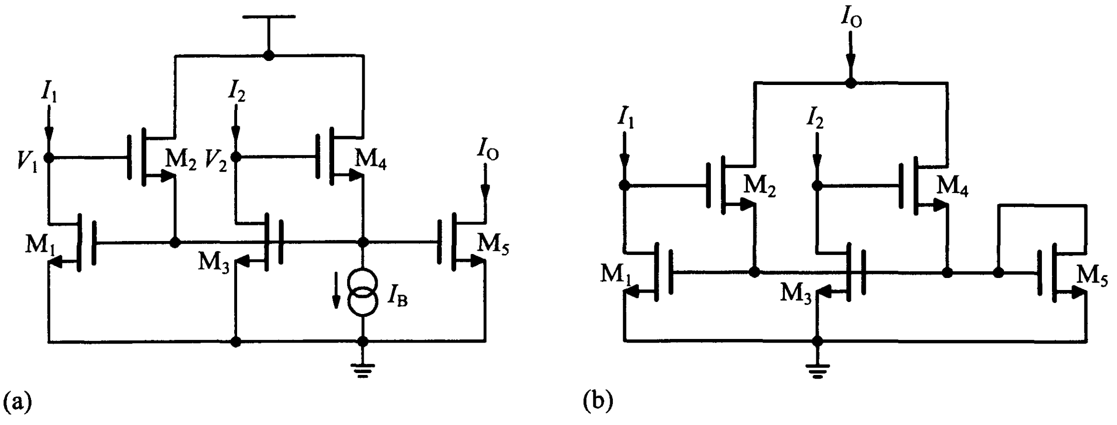


Fig. 4.13 Traditional asynchronous current-mode max circuits based on Lazzaro circuit [100].

maximum input current I_1-I_2. This current is recovered at the output by M_5. M_2 and M_4 are used as voltage followers, one of which establishes the common gate voltage which corresponds to the largest of I_1 and I_2. The other follower is OFF and therefore V_1 and V_2 can be used as digital outputs. The circuit of fig 4.13(b) is an improved version of the Lazzaro circuit reducing power dissipation [129]. It can also be seen as the connection of Wilson current mirrors which share the same output diode connected transistor M_5. Further modifications of this circuit have been proposed in order to improve matching accuracy of the asymmetrical nature of the Wilson current mirror [130]. However, despite their low complexity, WTA circuits of these types feature large voltage swings at their internal nodes, over which parasitic capacitors charge and discharge, resulting in very poor high frequency performance. Also, in order to accommodate these swings large voltage supplies are required, which make them unattractive for low power design.

Therefore, due to the nature of the algorithm and the required speed, a pure current-mode min/max circuit (without any associated large voltage drops) is a more suitable approach. This can be achieved by the use of a synchronous, mixed signal *replicating current comparator* (RCC) [105]. These circuits are mixed-signal using a simple form of a clocked latch incorporating positive feedback triggered by very small differential input currents. As a result, a fast decision circuit is achieved at the expense of extra clock signals and increased power dissipation. Since low complexity is a key feature here, in order to exploit the advantages of the analogue approach, the amount of clock signals and related digital circuitry should be minimal. Three different RCC circuits have been used for the CS operation in the various versions of the MFD and will be described in section 4.3.5.1.

The last stage of an ACSU is the memory which consists of two identical cells. One cell is acquiring new information fed from the CS circuit while the other is holding the previous decision for use in the feedback loop. Clearly, it is convenient to use a current-mode memory cell in order to perform all necessary processing in the current domain, without the need for current to voltage and vice versa conversions as discussed above. A current-mode memory cell can be realised using (SI) circuit techniques as in [11]. The detailed operation of these cells will be explained in section 4.3.5.2.

4.3.5.1 ACS Circuit

As it was discussed above, the addition in a current-mode ACSU is obtained very easily by connecting the various BMC branches to a low impedance node. In the following three circuits this addition in each ACSU input is obtained at the input of a PMOS high swing current mirror. There are ten current branches connected in each mirror. Nine branches come from the BMC (as it was discussed in section 4.3.4) and one branch constitutes the local feedback-loop carrying the value of the previous state metric.

ACS Circuit 1

Fig. 4.14 shows the schematic diagram of the ACS used in the first version of the chip. The circuit consists mainly of two PMOS current mirrors M_{11}-M_{13} and M_{15}-M_{18}, where the addition takes place as discussed above, and a low complexity RCC realising the max operation. The RCC consists of the simplest form of an NMOS latch (M_5-M_6), two NMOS high swing cascode mirrors (M_1-M_4 and M_7-M_{10}) and two switches (S_1-S_2). All switches in the circuit are denoted by the letter "S" in order to be discriminated from the rest of the transistors. For simplicity, the RCC input currents I_A and I_B represent the resulting $I_{AA,t} + I_{\Gamma(A),t}$ and $I_{CA,t} + I_{\Gamma(C),t}$ respectively.

The operation of the RCC is based on positive feedback entailed by the latch, which consists of two cross coupled inverters (M_5-M_6). S_1 and S_2 enhance the performance of the circuit by speeding up the decision of the latch. When F_{ACS} is high the positive feedback is disabled and the associated parasitic capacitors at nodes A and B are discharged through S_1 and S_2. When F_{ACS} goes low, these nodes start charging and the

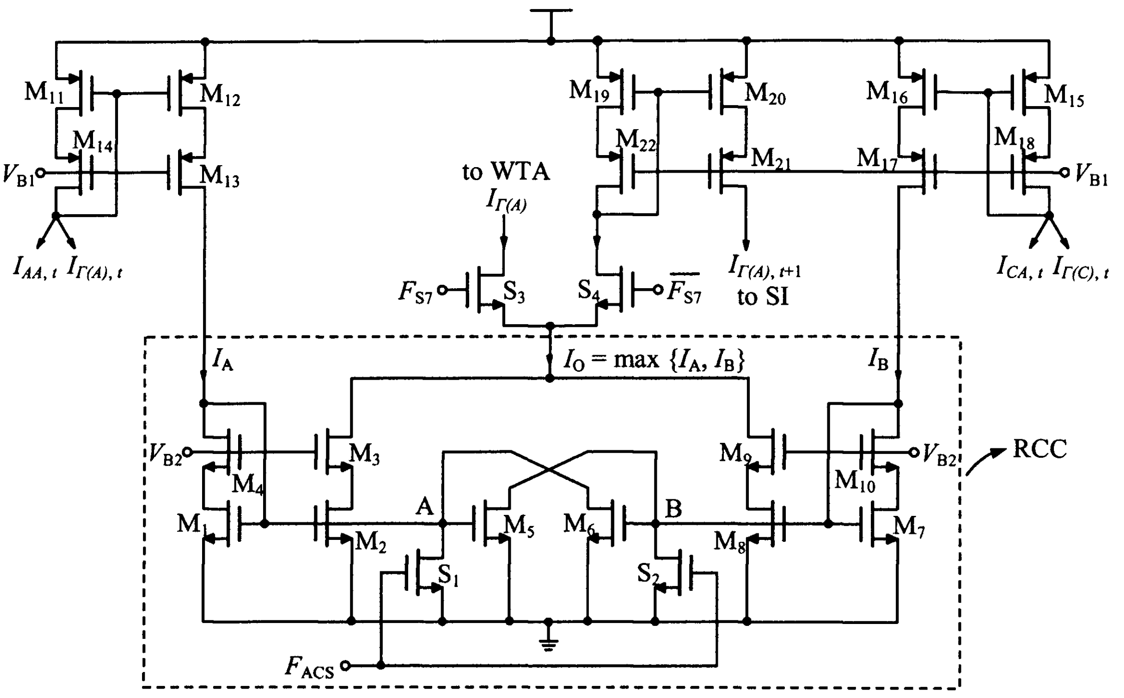


Fig. 4.14 ACS circuit 1 (for the upper or lower ACSU$_A$).

instant their voltages V_A and V_B reach V_t (i.e. the threshold voltage of M_5 and M_6) a competition is initiated which is rapidly amplified by the positive feedback. The largest of I_A and I_B results in the largest V_A or V_B respectively which eventually switches M_5 or M_6 on and M_6 or M_5 off. Hence, one of the RCC mirrors is off while the other one replicates the winning current into the output I_O. I_O is directed to the SI memory cell by means of S_4 and the mirror M_{19}-M_{22} for the first six ACS cycles (i.e. F_{S7} low). When F_{S7} is high, I_O carries the total path metric for the complete sliding block of $L = 10$ levels in the modified tree and therefore needs to be sent to the WTA block for final decision.

Table 4.5 shows the various transistor dimensions used in ACS circuit 1. The complete ACSU timing diagram for all ACS circuits is given in fig 4.20.

Table 4.5 Transistor dimensions for ACS circuit 1.

Transistor	W (μm)	L (μm)
M_1-M_2, M_5-M_8	4	1.6
M_3-M_4, M_9-M_{10}	4	1
M_{11}-M_{12}, M_{15}-M_{16}, M_{19}-M_{20}	7	1.6
M_{13}-M_{14}, M_{17}-M_{18}, M_{21}-M_{22}	7	1
S_1-S_4	2	0.8

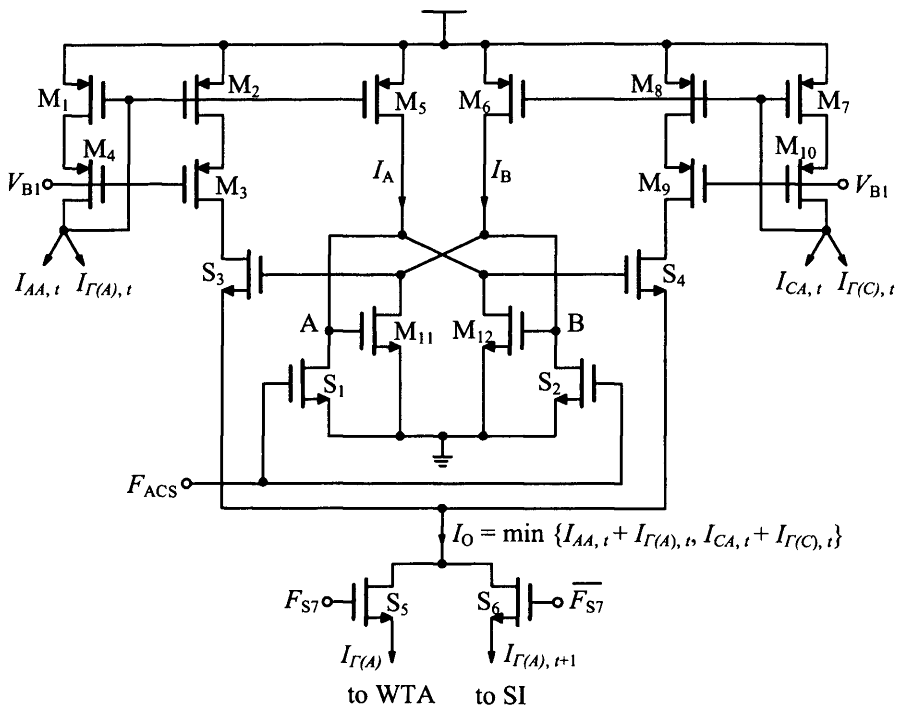


Fig. 4.15 ACS circuit 2 (for the upper or lower ACSU$_A$).

ACS Circuit 2

The ACS circuit of fig. 4.15 is used in the second version of the chip (see table 4.6 for information on transistor dimensions). It consists of a RCC, based on high gains of positive feedback using the NMOS latch of fig. 4.14. The difference here is the fact that nodes A and B are charged by the input currents, amplified by M_5 and M_6. When F_{ACS} goes low, the difference in V_A and V_B triggers the latch into a stable state. The nodes A and B settle on complimentary binary voltages which control switches S_4 and S_3 respectively. Therefore, the minimum of $I_{AA,t} + I_{\Gamma(A),t}$ and $I_{CA,t} + I_{\Gamma(C),t}$ is replicated to the output I_O. In this circuit, I_A and I_B are twice as large as the input currents in

Table 4.6 Transistor dimensions for ACS circuit 2.

Transistor	W (μm)	L (μm)
M_1-M_2, M_8-M_7	10	1.2
M_5-M_6	20	1.2
M_3-M_4, M_9-M_{10}	10	0.8
M_{11}-M_{12}	4	1.2
S_1-S_2	1.6	0.6
S_3-S_5	3.2	0.6
S_6	8	0.6

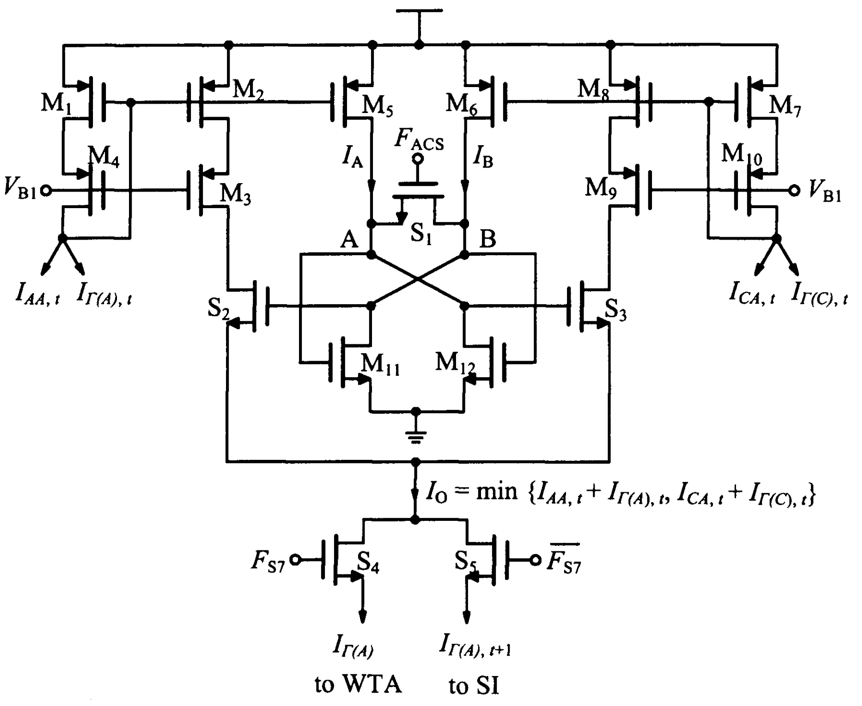


Fig. 4.16 ACS circuit 3 (for the upper or lower ACSU$_A$).

order to achieve higher resolution compared to the RCC of circuit 1. The need for a minimum operator in the second chip has also led to a much simpler ACS structure, which makes the use of extra mirroring in the input and output currents redundant (compare with circuit 1). However, the circuit of fig. 4.15 can also perform maximum operations, simply by interchanging the gate connections of S_3-S_4.

ACS Circuit 3

The ACS of fig. 4.16 is an improved version of circuit 2. Apart from the use of larger gate length to improve matching in the critical transistors (see table 4.7), a major issue

Table 4.7 Transistor dimensions for ACS circuit 3.

Transistor	*W* (μm)	*L* (μm)
M_1-M_2, M_7-M_8	12	3.6
M_5-M_6	24	3.6
M_3-M_4, M_9-M_{10}	16	0.6
M_{11}-M_{12}	6	1.8
S_1	6.4	0.6
S_2-S_4	3.2	0.6
S_5	8	0.6

imposed by circuit 2 was also resolved. The test structures used in the second chip proved that the resolution of the RCC of circuit 2 was inadequate compared with the simulated results. Further investigation in the layout led to identification of the cause, which will be explained next. The latch operation depends on the precise matching of the associated parasitic capacitances at nodes A and B. Any uneven track parasitics introduced on these nodes (i.e. due to asymmetrical layout), can result in erroneous decisions whereby the latch switches into the wrong state. This phenomenon is more pronounced on the latch of circuit 2, where high gains of positive feedback can rapidly amplify a small voltage difference. Due to the low gain of the latch in circuit 1, any uneven parasitics will just result in an extra delay, without any malfunction in the operation, as it will be seen next. This is rectified in circuit 3 by using a different approach to reset the latch. Switches S_1-S_2 in fig. 4.15 are replaced by S_1 in fig. 4.16. Having the switch in between the gates of M_{11}-M_{12} in circuit 3 has the following advantages. In the reset phase (F_{ACS} high) the latch transistors are always on (in saturation), which increases the speed of operation since nodes A and B do not constantly charge and discharge. This in turn reduces the latch sensitivity to uneven parasitics as these nodes are pre-charged the instant F_{ACS} goes low. Matching of the latch transistors is also enhanced then, since both M_{11}-M_{12} are in saturation before the competition starts. The use of one clock (that drives S_1) improves dynamic coupling between digital signals and the critical nodes making the circuit less sensitive to physical layout too.

The performance characteristics for the three different ACS circuits are summarised in table 4.8. Circuit 1 may seem to outperform the others, however, it features practical weaknesses which degrade the overall performance. The delay in the output

Table 4.8 Simulated performance characteristics for the various ACS circuits.

Parameter	*ACS Circuit 1*	*ACS Circuit 2*	*ACS Circuit 3*
Operator	MAX	MIN	MIN
Worst delay (ns)	17	17	44
Average Power (μW)	300	333	350
Resolution* (μA)	0.5	1	1
Input range (μA)	10-30	10-40	5-40

*Monte Carlo simulations

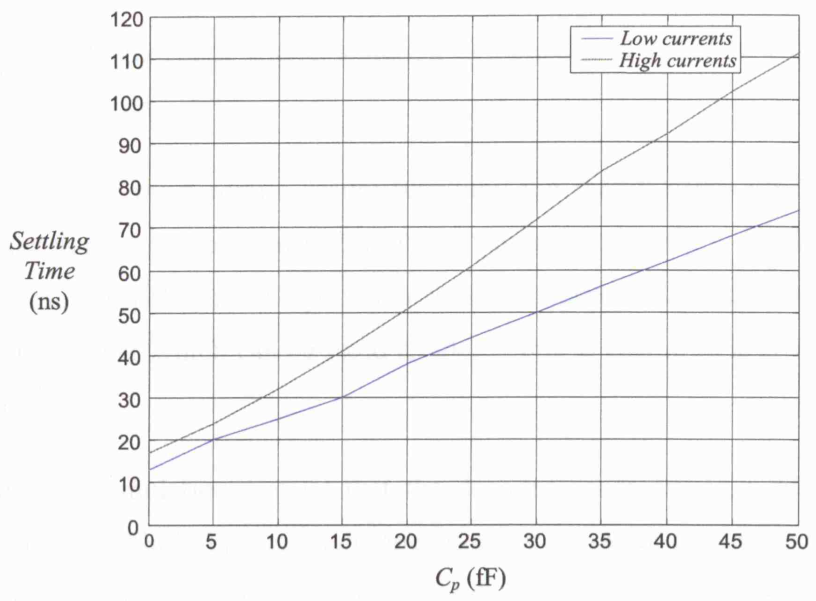


Fig. 4.17 Settling time against parasitic imbalance C_p at nodes A and B of ACS Circuit 1.

is signal dependent and increases with the input currents. This in particular is an issue when the difference between the inputs is smaller than the quoted resolution. Figure 4.17 shows a graph of the settling time against the parasitic imbalance C_p between nodes A and B for $|I_A - I_B| = 1\mu A$. The use of the PMOS input current mirrors in circuit 1 act as buffers which reduce C_p below 50fF. However, when this circuit is used multiple times in the decoder its speed is determined by the slowest response. Even the most careful and symmetrical layout can result in malfunction, since the extracted parasitic capacitances used in post layout simulations are never identical to those encountered in practice. Therefore, circuit 1 features a very slow response especially at high currents (i.e. winning currents) and increased complexity. Due to its low positive feedback gain the minimum input difference resolved can be as low as 0.5μA at the cost of an extra delay. However, in order to achieve the delay shown in table 4.8, a resolution of 1μA was chosen.

On the other hand, circuit 2 features a very high gain incorporated by amplification which rapidly switches the latch fully to complimentary binary states. This in turn forces the latch to be almost independent of the signal levels. However, as it was described above, a very small C_p in the order of few fF impairs the operation resulting in malfunction. Also, another disadvantage of circuit 2 is the average power consumption which is proportional to the width of F_{ACS} duty cycle. When F_{ACS} is high, the amplified input currents I_A and I_B flow through S_1-S_2 increasing static power. Therefore, the decoder power consumption depends on F_{ACS}, which can increase clock generator implementation complexity in order to be kept low at all operational speeds.

Circuit 3 is the most robust design resolving the major issues discussed above. In particular the latch operation is completely insensitive to $C_p < 100\text{fF}$, which can be easily achieved with careful layout. However, the increased L (see table 4.7) results in a slower response which can be fixed to about 50ns to ensure I_O is always settled to its final value.

System-level simulations reveal that the decoder needs an ACS resolution of 1μA over a range of 20μA for maximum operations or 10μA for minimum operations. This translates to a resolution of about 4.3 and 3.3 equivalent-bits respectively. These simulations were performed in a similar manner as in section 3.5.3 using Simulink®. Hence, from table 4.8 it becomes apparent that circuits 2 and 3 can operate well within the required range.

4.3.5.2 SI Memory Cell

SI memory cells are used in place of SC circuits in analogue current-mode signal processing systems [23]. Their operation depends on the gate to source parasitic capacitance C_{GS} of a MOS transistor, eliminating the use of bulky and expensive capacitors. Since they are based on charge transfers between small capacitances, their main drawback is the charge injection error induced by the associated clock signals through switches. The latter effect is known as clock feed-through and can severely affect the operation of SI circuits [23]. However, a resolution of about six equivalent-bits can be easily achieved without the need of complex compensation schemes [103]. Therefore, and since the required resolution in the ACSUs is less than 5 equivalent bits, the same SI cells as in [11] are used in all three versions of the chip, which will be discussed next.

Fig. 4.18 shows the SI cell used in the decoders. It is essentially an extended version of a high swing cascode SI memory featuring two outputs. It is designed this way in order to be directly compatible with the PMOS input mirrors of the ACS circuits it feeds. The operation of this cell is described as follows. When F_1 is high, M_1 and M_3

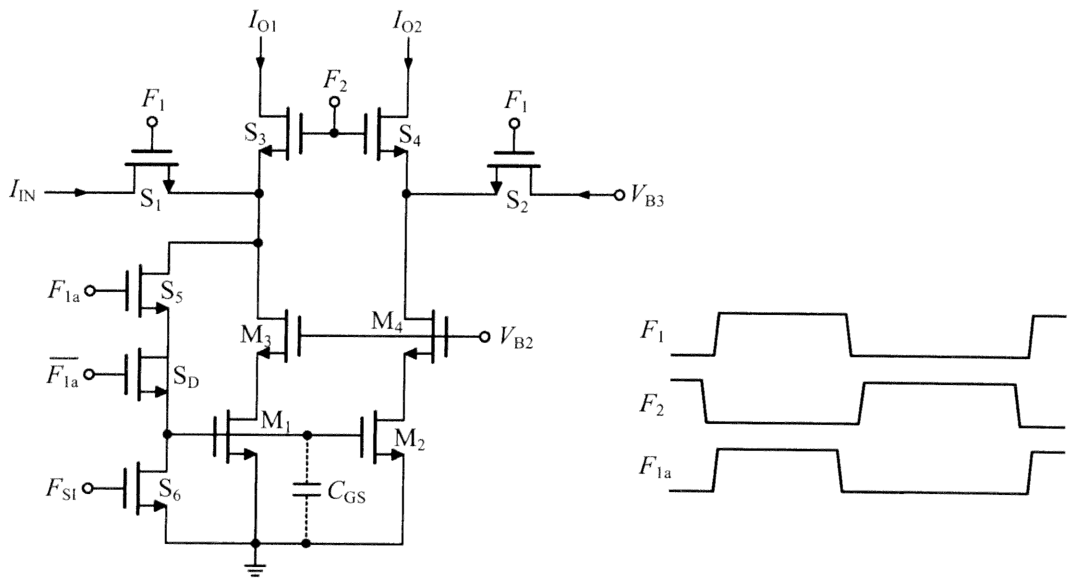


Fig. 4.18 SI memory cell and timing diagram.

form the input diode of the mirror M_1-M_4 (F_{1a} is high). The instant F_1 goes high, I_{IN} is charging C_{GS} through S_1 to a certain voltage V_{GS1}. C_{GS} is the total parasitic capacitance on the common gates of M_1-M_2, which is equal to $2 \cdot C_{GS1}$ assuming these transistors are matched and C_{GS1} is the gate to source capacitance of M_1 and M_2. At the same time the mirrored output is directed to a fixed bias V_{B3} through S_2. When F_2 is high, the charge stored on C_{GS} retains V_{GS1}, which in turn reproduces the input current into the output I_{O1} and I_{O2} through S_3 and S_4, respectively. Hence, this cell is feeding two different ACSUs when F_2 is high. Therefore, the use of two identical cells of the type shown in fig. 4.18 is necessary in each ACSU, which work in a "ping pong" fashion. These cells need to be reset at the beginning of every decoding cycle in order to carry no metric information over from the previous sliding block. This is achieved by S_6 and pulse F_{SI}.

Note that the above covers a minimal description of how the practical cell of fig. 4.18 actually operates. The timing diagram in the same figure suggests that S_5 switches off before S_1 to ensure that I_{IN} is not interrupted during sampling. Also, F_1 and F_2 are non-overlapping clocks in order to avoid interference between acquired input and held output. Furthermore, the use of dummy S_D compensates for the induced clock feed-through errors arising from the use of S_5 and F_{1a} [103]. Charge transfers from the associated parasitics of S_5, which can affect V_{GS1} when F_{1a} goes low, are absorbed by S_D. In fact, and providing that the rise and fall times of the clock signals are identical

S_D should be made half the size of S_5 for complete cancellation of the *signal-dependent* error [103] (excluding channel length modulation errors). Any *signal-independent* error simply causes a DC offset to appear in all the ACSUs outputs, which has no effect on the performance of the decoder. As it was shown in [103], in a maximum ACS operation suitable for a Viterbi decoder, the choice of a larger S_D geometry results in path metric expansion, which favours the more likely paths (paths with large metrics). However the opposite effect, called path metric compression, results in a significant loss in *CG*. In the case of a hard-decision MFD where the path metric range is limited to only 3 to 4 equivalent-bits, the effect of metric expansion or compression is negligible, even for a maximum ACS operation. This was verified for both MIN and MAX MFDs, using similar Simulink® modelling as in section 3.5, for an error of up to 15% of each ACSU output signal. This error is very unrealistic since circuit level simulations indicate that a value of less than 3% is to be expected in practice.

The complete memory circuit consisting of two SI cells is shown in fig. 4.19. Table 4.9 shows the transistor dimensions for the various versions of this circuit. Minor

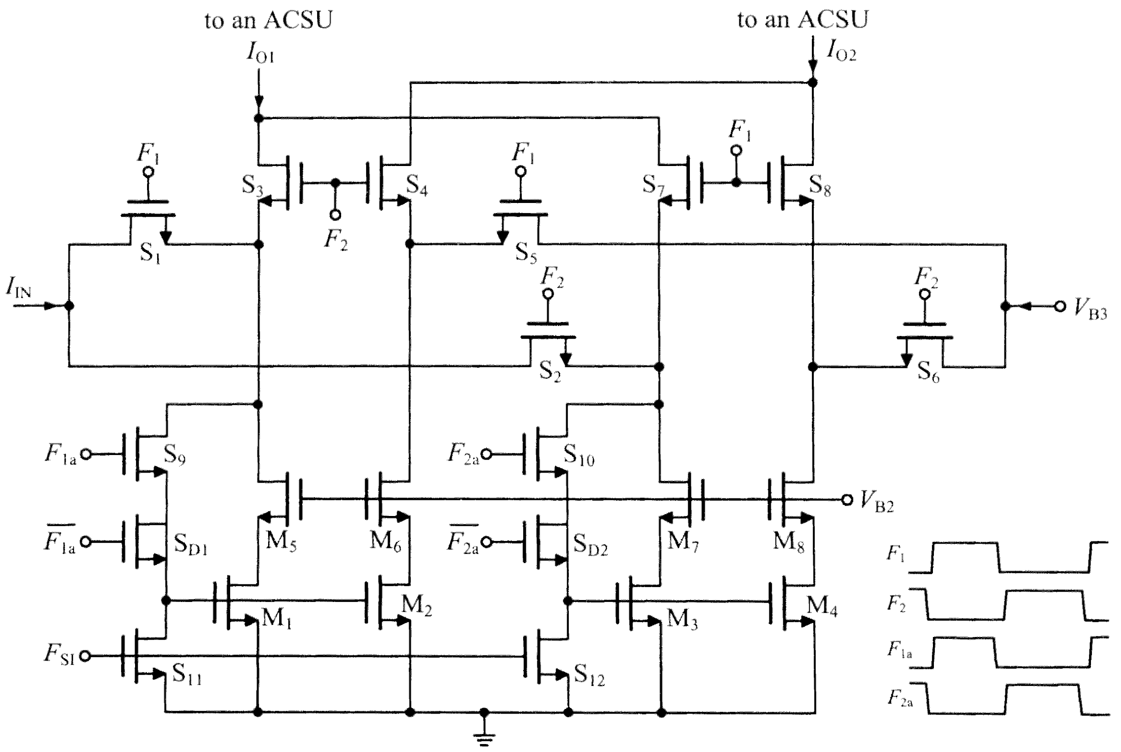


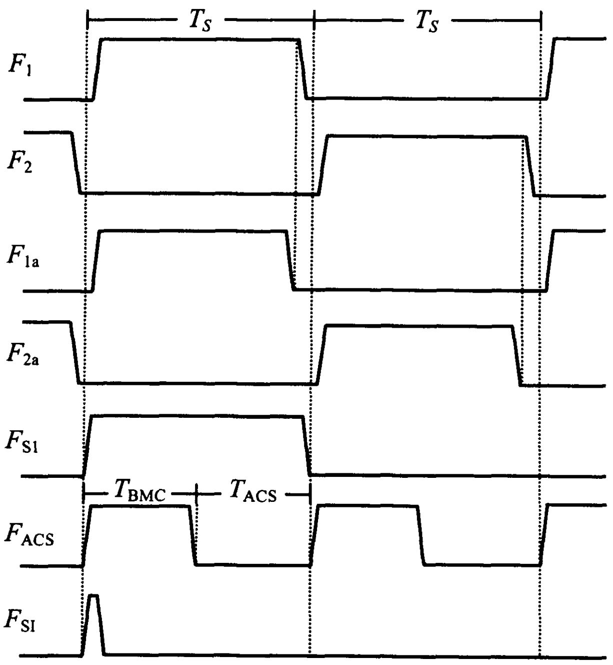
Fig. 4.19 The complete SI memory circuit used in all versions of the MFD and timing diagram.

Table 4.9 Transistor dimensions for the SI cell of fig. 4.19 used in various versions of the MFD.

Version 1			Version 2			Version 3		
Transistor	W (μm)	L (μm)	Transistor	W (μm)	L (μm)	Transistor	W (μm)	L (μm)
M_1-M_4	10	1.2	M_1-M_4	8	1.2	M_1-M_4	12	3.6
M_5-M_8	10	1.2	M_5-M_8	8	1.2	M_5-M_8	6	0.6
S_1-S_8	2	0.8	S_1-S_2	3.2	0.6	S_1-S_2	3.2	0.6
S_9-S_{10}	4	0.8	S_3-S_{10}	1.6	0.6	S_3-S_{10}	1.6	0.6
S_{11}-S_{12}	2	0.8	S_{11}-S_{12}	0.8	0.6	S_{11}-S_{12}	2	0.6
S_{D1}-S_{D2}	2	0.8	S_{D1}-S_{D2}	0.8	0.6	S_{D1}-S_{D2}	0.8	0.6

dimension changes occur between versions 1 and 2 to accommodate the different technologies used. However, in version 3 a much larger gate length L is used in all mirror transistors to ensure better performance. In order to reduce the consequent delay, the aspect ratio (W/L) of the cascode transistors M_5-M_8 is increased by a factor of three, which can be shown to improve the settling time of the mirrors [23]. Versions 1 and 2 attribute a delay of 6ns and 7ns, respectively, and version 3 attributes a delay of 20ns to the total required ACS time T_{ACS}.

Fig. 4.20 shows the precise timing diagram for a complete ACSU. As discussed



T_{ACS} (ns)		
Version 1	Version 2	Version 3
30	24	64

Fig. 4.20 ACSU complete timing diagram and T_{ACS} for various versions of the MFD.

previously (see section 4.3.3), T_{ACS} and T_{BMC} determine the width of T_S and therefore the decoding speed $T_D = 8 \cdot T_S$. Fig 4.20 also shows the total required T_{ACS} for all versions of the MFD, allowing time for both ACS and SI circuits. Version 1 has a higher T_{ACS} than that of version 2, mainly because of the extra replication of the I_O through a PMOS mirror before it reaches the SI circuit (see fig. 4.14). F_{SI} pulse width can be as small as 10ns to ensure that the memory cells are reset before the falling edge of F_{ACS}. Note here that since F_{S7} is twice as wide as F_{S1}-F_{S6} (see section 4.3.2), when F_{S7} is high, F_1, F_2, F_{1a} and F_{1b} need to be twice as wide to avoid any signal interruption when the ACS block output is processed by the WTA. Also, at that time F_{ACS} should have a missing pulse for the same purpose.

The complete ACS block (upper and lower) is shown symbolically in fig. 4.21 illustrating the internal connectivity of the units. Note that this resembles the sequential steps on a trellis diagram (i.e. see fig. 3.5). For simplicity, the connection to clock and bias signals in each ACSU is not shown.

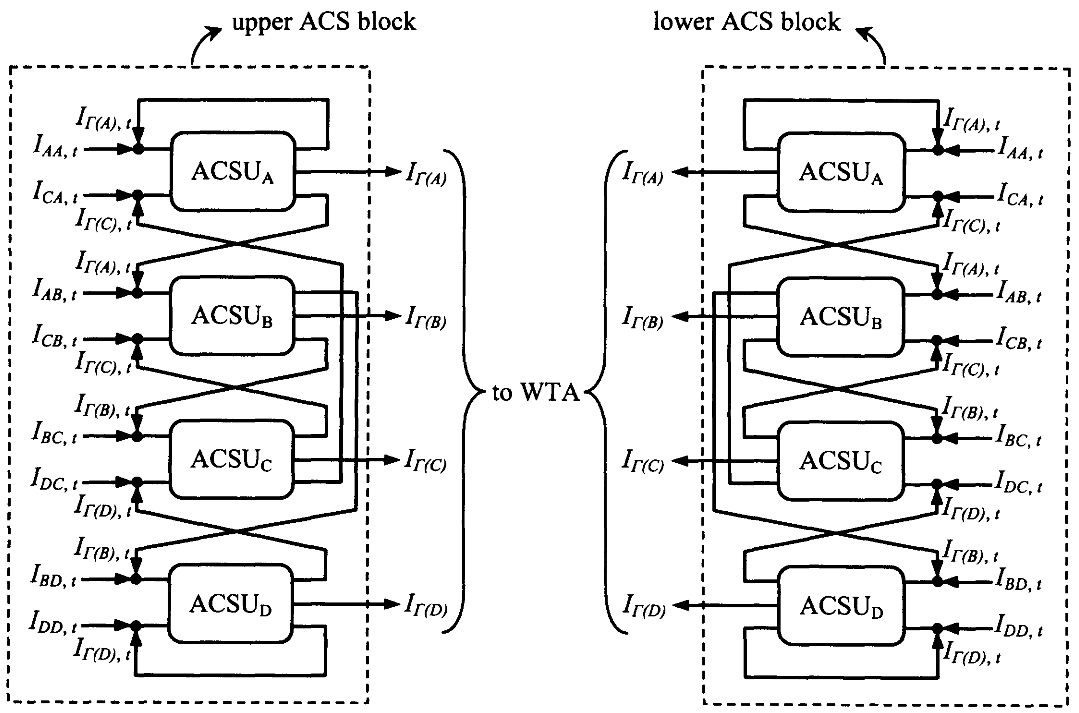


Fig. 4.21 The complete upper and lower ACS blocks illustrating the ACSU connectivity.

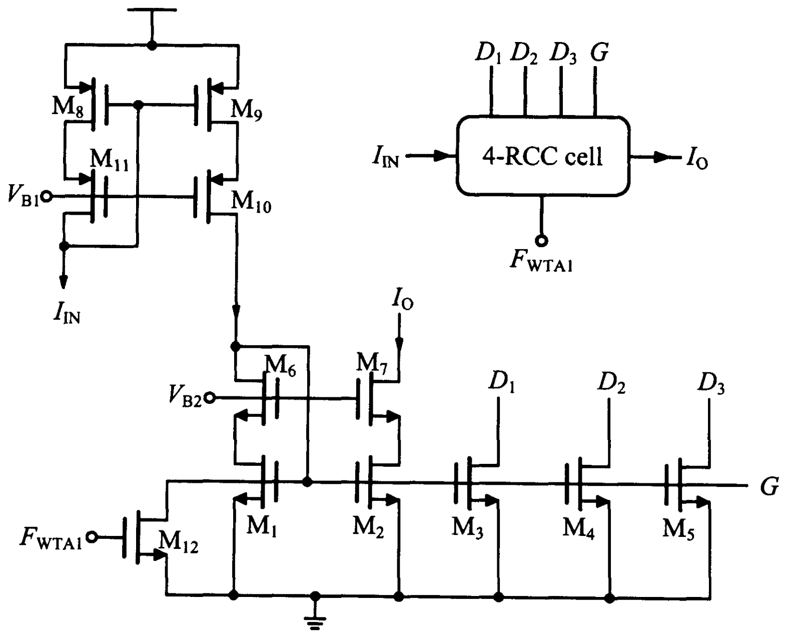
4.3.6 WTA Block

WTA circuits are very similar in operation to the max circuits described in the previous section. The WTA differs from the CS circuit used in a MFD in the following two aspects. First, it has multiple inputs (2^K) and second, only identification of the largest of its inputs is required and no replication of its winner is necessary. In general, a WTA circuit consists of multiple cells (i.e. one cell per input) with binary outputs indicating whether the cell is ON or OFF. Its operation suggests that only the winner cell is ON while the remaining cells are inhibited. Similarly, in a *loser-take-all* (LTA) circuit the loser cell corresponding to the minimum of its inputs is ON while the remaining cells are OFF.

For reasons explained in section 4.3.5 and since the ACS block is entirely current-mode, a WTA circuit which features current inputs is necessary. There are several asynchronous WTA circuits reported in the literature mainly based on modifications of the Lazzaro circuit [100], [129]-[131]. These circuits are asynchronous and feature simultaneous multiple input (i.e. all inputs are applied in parallel) interface. However, in addition to the disadvantages when used as 2-input max circuits (see section 4.3.5), their multiple input configuration in a WTA significantly degrades the performance. Both resolution (due to poor transistor matching) and speed deteriorate at a rate proportional to the number of inputs [101]. Other asynchronous current-mode WTA architectures can be found in [132]-[135]. In [135], a current-mode version of the common source circuit described in section 4.3.5 (see fig. 4.12) is realised, based on the flipped voltage follower [136]. This topology essentially eliminates the large power supply requirements of the Lazzaro type circuits while reducing the corner error of the common source circuit. Extended to multiple inputs, high resolution with very low power supplies can be achieved; however the low speed of its asynchronous operation remains. Also, the WTA of [132] consists of an array of simple current mirrors connected in positive feedback-loops, incorporating very low gain, and therefore features very low settling times. In addition, its precision depends on the global matching of all $M^2 + M$ transistors (where M is the number of inputs), which becomes an issue for $M > 4$. The latter constraint is resolved in [133], where no performance characteristic relies on global matching but it still suffers long delays. A novel synchronous WTA architecture is presented in [134], which features very high speed and precision. This is achieved by employing a slightly complex but very high gain feedback mechanism (i.e. inhibitory and local excitatory feedback) which is based on the computation of the average of its current inputs. However, compared with other synchronous WTA circuits, the complexity of this circuit per input cell cannot be justified when moderate precision is required as in a MFD. These other circuits are based on the RCC described in section 4.3.5.1, extended to include *M* inputs as it will be shown next.

4.3.6.1 WTA Circuit 1

Fig. 4.22 shows a 4-input RCC cell (4-RCC) used for the construction of the WTA circuit in the first version of the chip. M_8-M_{11} form the input mirror. The NMOS mirror of this cell has four outputs, three of which (D_1-D_3) are used to form six latches in the complete 4-RCC. This is achieved by interconnecting four 4-RCC cells in such a manner that each of D_1-D_3 in a cell connects to the common gate of the other cells, respectively. The complete WTA circuit is shown in fig. 4.23. It consists of two 4-RCC circuits corresponding to the upper and lower ACS blocks and a CMOS latch identical to the one used in fig. 4.2. I_U and I_L correspond to the largest of the upper and lower $I_{\Gamma(A)}$, $I_{\Gamma(B)}$, $I_{\Gamma(C)}$ and $I_{\Gamma(D)}$ respectively. The final WTA decision is determined by the CMOS latch where the comparison of I_U and I_L takes place.



Transistor	W (μm)	L (μm)
M_1- M_5	4	1.6
M_6-M_7	4	1
M_8-M_9	7	1.6
M_{10}-M_{11}	7	1
M_{12}	2	0.8

Fig. 4.22 4-RCC cell used in WTA Circuit 1 and its symbol.

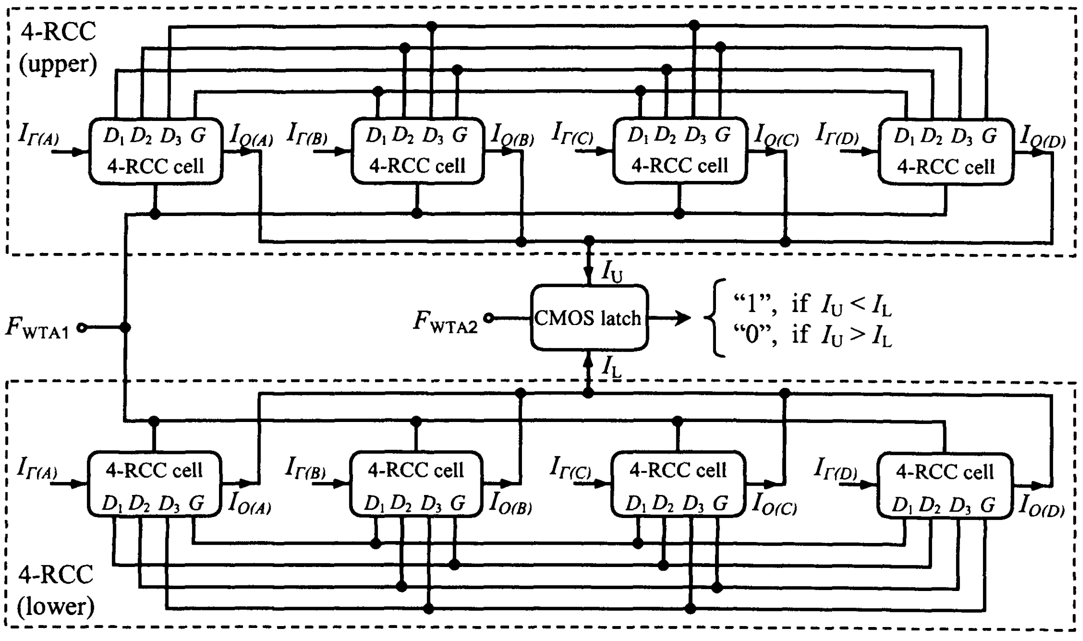


Fig. 4.23 WTA circuit 1 constructed using two 4-RCCs and a CMOS latch.

The 4-RCC works in exactly the same manner as the 2-input RCC (2-RCC) version of fig. 4.14. When F_{WTA1} goes low, a competition is initiated between all twelve latch transistors and is accelerated by means of positive feedback. As a result, all loser cells turn off and the winner mirror replicates the maximum current to its output.

4.3.6.2 WTA Circuit 2

Although the 4-RCC described above can be modified to perform minimum operations using De Morgan's law as in [132], it involves extra mirroring which increases power and complexity and degrades performance due to mismatch. An alternative to processing all the current inputs in parallel is the use of 2-RCCs in a tree-structure [101]. The multi-stage nature of this approach may seem bulky and slow, however when the number of inputs is small, the performance is not greatly impaired.

In order to construct the complete WTA* circuit 2 two different types of 2-RCC circuits and a CMOS latch are used. The first type of RCC will be denoted as NMOS 2-RCC and the second as PMOS 2-RCC. The WTA operation is arranged in three different stages or levels. Since the outputs from the ACS circuit 2 (see fig. 4.15)

* Note that, although WTA circuits 2 and 3 perform minimum operations and should frankly be called LTA, they will be referred to as WTA in order to comply with fig. 4.1. This is often used in the literature when the minimum of a set of variables is pre-defined as the winner.

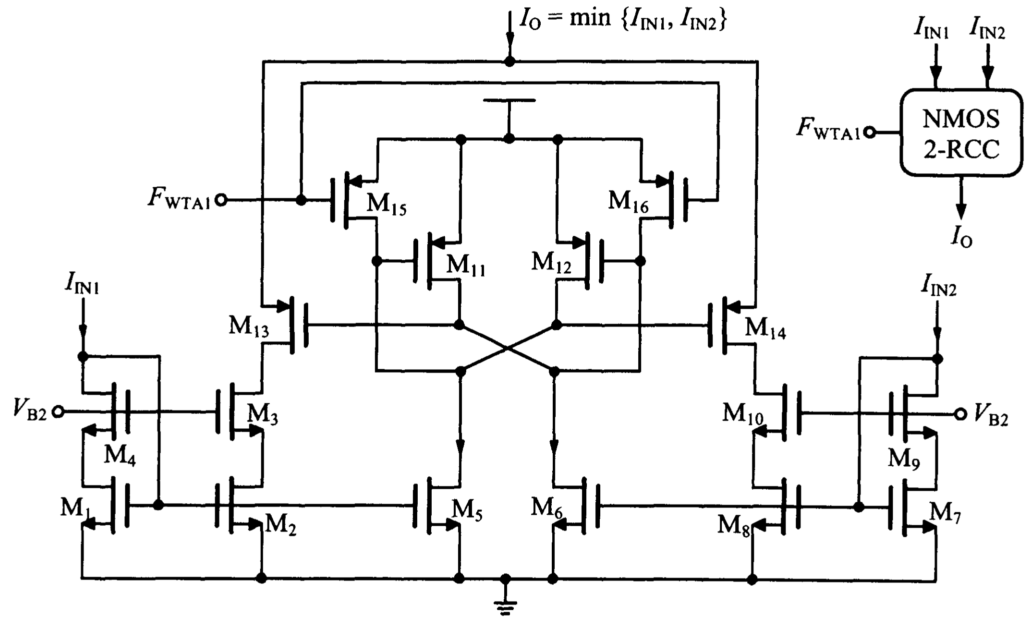


Fig. 4.24 NMOS 2-RCC used in WTA Circuit 2 and its symbol.

come from a PMOS mirror, the first level of WTA circuit 2 tree consists of four NMOS 2-RCCs. The second level consists of two PMOS 2-RCCs and the third level is a CMOS latch identical to that of fig. 4.23. Using this structure, redundant current replications are avoided reducing power and complexity.

Fig. 4.24 shows the NMOS 2-RCC. It consists of mirrors M_1-M_4 and M_7-M_{10}, current amplifiers M_5-M_6, a PMOS latch M_5-M_6 and switches M_{13}-M_{16}. This circuit works in exactly the same principle as that of fig. 4.15. In fact, the NMOS 2-RCC is the inverted version of the ACS circuit 2. The transistor dimensions are given in table 4.10.

The PMOS 2-RCC is shown in fig. 4.25. This is essentially the ACS circuit 2 of fig. 4.15. Table 4.11 shows the various transistor dimensions.

Table 4.10 Transistor dimensions for the NMOS 2-RCC of fig. 4.24.

Transistor	W (μm)	L (μm)
M_1-M_2, M_7-M_8, M_{11}-M_{12}	8	1.2
M_5-M_6	16	1.2
M_3-M_4, M_9-M_{10}	8	0.8
M_{13}-M_{14}	3.2	0.6
M_{15}-M_{16}	0.8	0.6

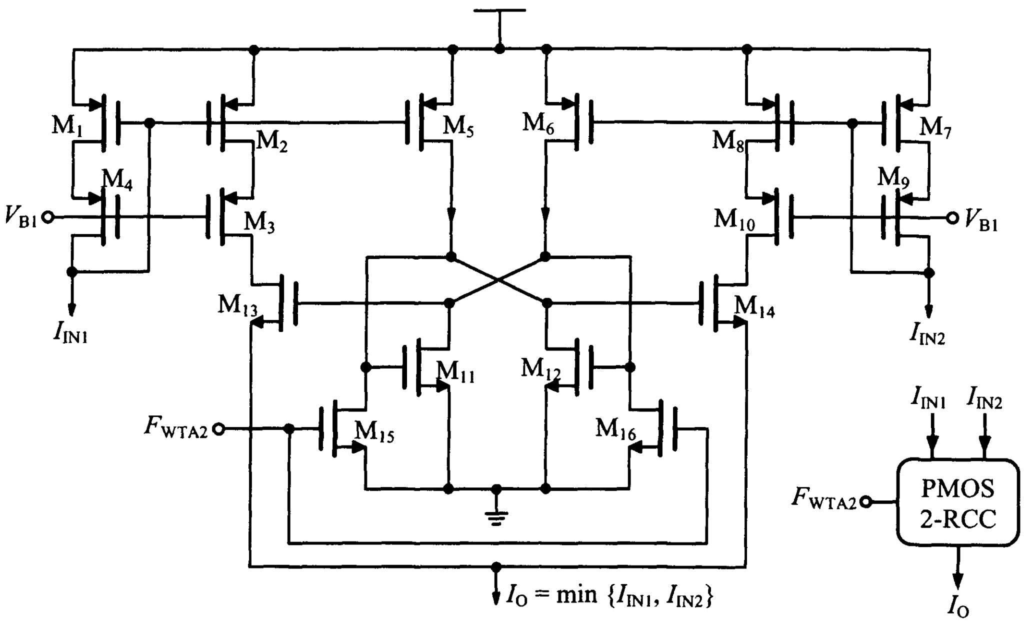


Fig. 4.25 PMOS 2-RCC used in WTA Circuit 2 and its symbol.

Table 4.11 Transistor dimensions for the PMOS 2-RCC of fig. 4.25.

Transistor	W (μm)	L (μm)
M_1-M_2, M_7-M_8	10	1.2
M_5-M_6	20	1.2
M_3-M_4, M_9-M_{10}	10	0.8
M_{11}-M_{12}	4	1.2
M_{13}-M_{14}	3.2	0.6
M_{15}-M_{16}	1.6	0.6

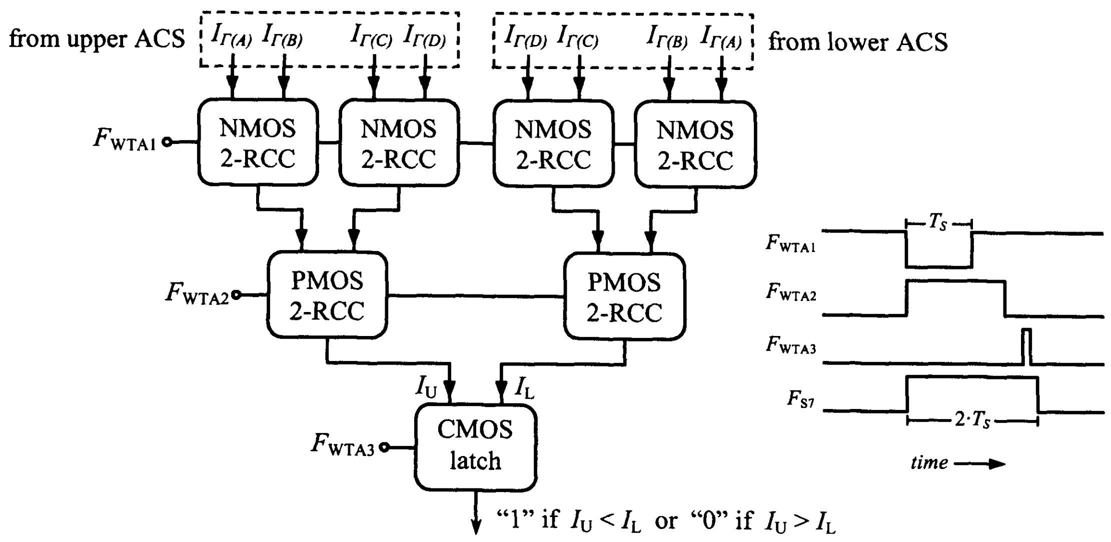


Fig. 4.26 WTA circuit 2 constructed using four NMOS and two PMOS RCCs and a CMOS latch.

The complete WTA circuit 2 and the timing waveforms for the reset pulses are shown in fig. 4.26. The RCC competition of the first level is initiated by the rising edge of F_{WTA1} (since M_{15} and M_{16} in fig. 4.24 are PMOS switches), the time instant the ACS output currents are settled at the seventh sequential cycle. This is exactly $T_S = T_{BMC} + T_{ACS}$ after the rising edge of F_{S7}. Allowing for all the NMOS RCCs to output a decision, the competition of the second level initiates another $T_{ACS} = T_S/2$ after that (falling edge of F_{WTA2}). Finally, F_{WTA3} steers the latch to a digital output corresponding to a decoded bit.

4.3.6.3 WTA Circuit 3

Similarly, the WTA circuit 3 is constructed using a 3-level network of RCCs and a CMOS latch identical to that shown in fig. 4.26. The only difference here is based on the difference between ACS circuit 2 and 3 (see section 4.3.5.1). That is, the PMOS RCC is identical to the RCC of ACS circuit 3 (see fig. 4.16) and the NMOS RCC is its inverted version. Therefore, the RCCs used in the WTA circuit 3 differ structurally from the RCCs of fig. 4.24 and 4.25 only in the way their latch is reset (i.e. switches M_{15} and M_{16} are replaced by a single switch in between the input/output nodes of the latch). Fig 4.27 shows the NMOS RCC used here. Also the associated transistor dimensions are shown in table 4.12.

Performance characteristics for all three WTA circuits are summarised in table 4.13.

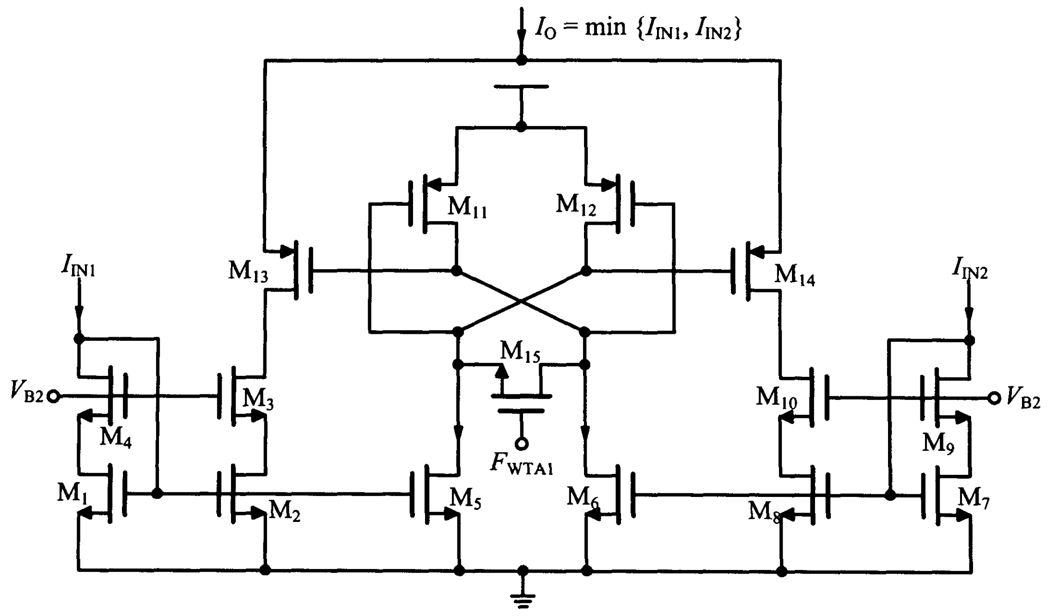


Fig. 4.27 NMOS 2-RCC used in WTA Circuit 3.

Table 4.12 Transistor dimensions for the NMOS 2-RCC of fig. 4.27.

Transistor	*W* (μm)	*L* (μm)
M_1-M_2, M_7-M_8	12	3.6
M_5-M_6, M_{11}-M_{12}	24	3.6
M_3-M_4, M_9-M_{10}	6	0.6
M_{13}-M_{14}	6.4	0.6
M_{15}	12	0.6

Table 4.13 Simulated performance characteristics for the various WTA circuits.

Parameter	*WTA Circuit 1*	*WTA Circuit 2*	*WTA Circuit 3*
Operator	MAX	MIN	MIN
Worst delay (ns)	38	52	154
Average Power (mW)	1.25	1.8	1.8
Resolution* (μA)	2	1	1
Input range (μA)	10-30	10-40	5-40

*Monte Carlo simulations

Although circuit 1 outperforms the rest in speed and power consumption, Monte Carlo results show that it is vulnerable to mismatch errors. This is to be expected as global matching of all latch transistors is required in order to avoid malfunction. On the other hand, in the tree structure of circuits 2 and 3 local matching in each RCC is adequate and easy to achieve in practice. However, the latter circuits suffer the additional delay of another processing stage. Nevertheless, the comparison between all three circuits is not fair as they are implemented using different CMOS processes and different transistor sizing strategy. Ultimately, a comparison of the complete decoders will be made taking into account these differences.

4.3.7 Feedback Network Block

The output decision of the WTA is processed by the feedback network block prior to being fed back to the BMC. The processing involves storing the decoded output for two decoding cycles as discussed in section 3.4.2, and mainly reproducing the variable code bits $s_{1,t}$, $s_{2,t}$, $s_{1,t+1}$, $s_{1,t+2}$, as described in section 4.3.4 (see fig. 4.7). The code bits are generated using a single gate, without the need to reconstruct the

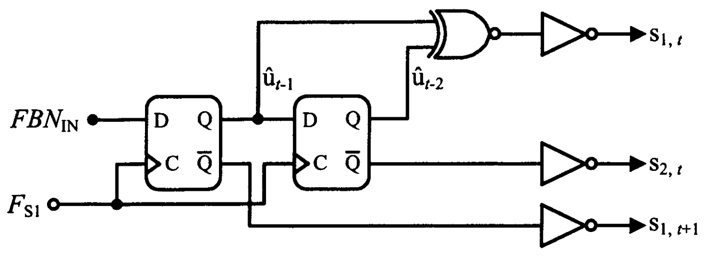


Fig. 4.28 Feedback network circuitry.

complete encoder in the system. This is explained as follows. Assuming $\hat{u}_{t-1}$ and $\hat{u}_{t-2}$ are the previous decoded bit and the one before that respectively, the contents of the encoder corresponding to the first three levels of the modified tree are according to fig. 3.9. Therefore, the related code bits are

$$s_{1,t} = \hat{u}_{t-2} \oplus \hat{u}_{t-1} \oplus 0 = \hat{u}_{t-2} \oplus \hat{u}_{t-1} \tag{4.3}$$

$$s_{2,t} = \hat{u}_{t-2} \oplus 0 = \hat{u}_{t-2} \tag{4.4}$$

$$s_{1,t+1} = \hat{u}_{t-1} \oplus 0 \oplus 0 = \hat{u}_{t-1} \tag{4.5}$$

$$s_{2,t+1} = \hat{u}_{t-1} \oplus 0 = \hat{u}_{t-1} = s_{1,t+1} \tag{4.6}$$

The above equations show that only one XOR gate is necessary in order for the feedback network to generate the required information for the variable states of the modified tree.

The complete feedback network circuit is shown in fig. 4.28. Two D-type flip flops are used for storing the previous decoded outputs controlled by F_{S1}, which samples the WTA output $\hat{u}_t$ at the beginning of the next decoding cycle. The inverters incorporate very large geometry transistors and are used to buffer the output signals.

4.3.8 Clock Generator Block

The MFD is a synchronous sequential system controlled by a number of clock signals, which ideally need to be generated in the same chip. Although this was not the case in the first version of the decoder, a clock generator was designed for the following two versions. A brief description of the main sub blocks used will follow next. Most gates and memory elements are standard cells found in the library of the corresponding technology, which for the two latest versions is AMS 0.6μm CMOS technology.

Both clock generator blocks used in the two versions of the MFD are very similar. They consist of standard gates and DFFs in order to accommodate the various clock waveforms described in the previous sections. All clock signals are derived from two master clocks, F_{MS1} and F_{MS2}, generated externally. F_{MS1} is 32 times faster than the decoding cycle $f_{\mathrm{MS1}} = 32 \cdot f_{\mathrm{D}}$ and $f_{\mathrm{MS2}} = 64 \cdot f_{\mathrm{D}}$. In the following, the combinatorial logic used for the generation of most clock signals is omitted while the focus is on the generation of the main clocks which the former are derived from.

Fig. 4.29 shows the ring counter used for the generation of pulses F_{S1}-F_{S7}. Eight DFFs are cascaded to form a shift register with feedback. When F_{RS} is high all DFFs are reset except the first which is set to high. When F_{RS} goes low, a pulse of width $2/f_{\mathrm{RC}}$ appears on Q_1 which is sequentially shifted in the register. Q_7 is twice as wide as Q_1-Q_6 in order to generate F_{S7} (see fig 4.6). Q_1-Q_7 must be converted to non-overlapping pulses since the processing of each sliding block in the MFD should be independent of others. The way to do this is by using a non-overlapping generator circuit of the type shown in fig. 4.30. The inverters following $Q_{i\text{-}1}$ have longer than the minimum gate lengths and are used to delay the signal. Also, the inverters following the NAND gate are used for buffering and feature exponentially increasing geometries in order to supply large currents to their consequent loads. All pulses $F_{\mathrm{S}i}$ are generated using a Q_i and a $Q_{i\text{-}1}$, except F_{S1}, where Q_7 is used instead of $Q_{i\text{-}1}$.

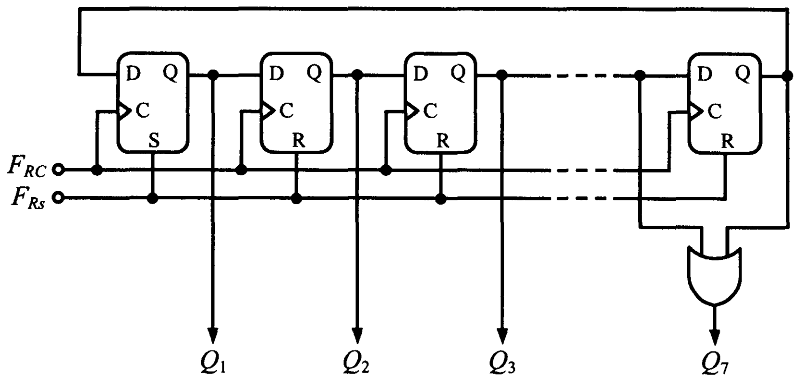


Fig. 4.29 Ring counter used in clock generator circuit.

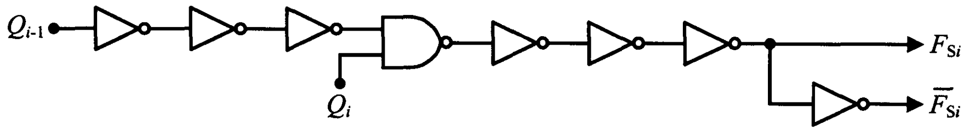


Fig. 4.30 Non-overlapping generator for pulses F_{S1}-F_{S7} (see fig. 4.6).

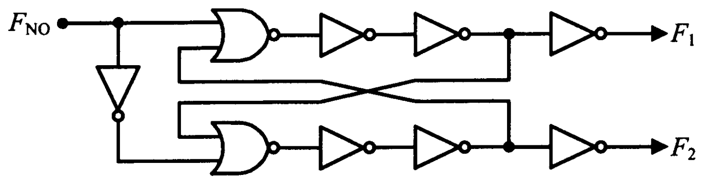


Fig. 4.31 Non-overlapping generator for pulses F_1 and F_2 (see fig. 4.20).

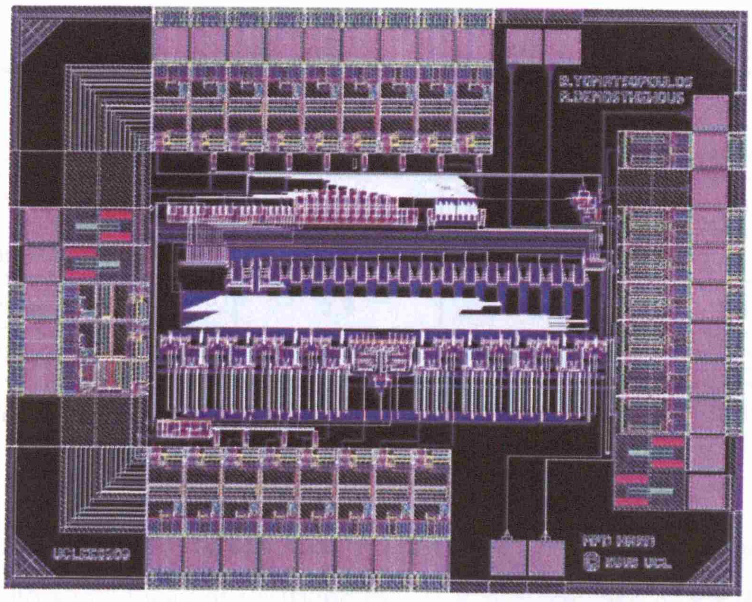
F_{MS1} is divided internally up to sixteen times in order to produce various pulses, one of which is F_{RC}. The division by two is easily done by the use of a DFF with its inverted output connected to its input. Any DFF used in the clock generator is connected to the same reset F_{Rs}, which is controlled manually by an external switch. This is not only used in the case of a *stuck* fault but also makes testing of the decoder chips easier.

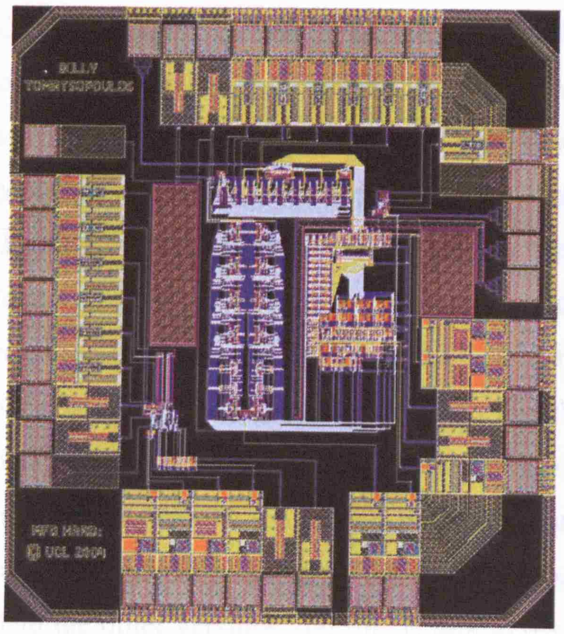
Finally, the clocks F_1 and F_2 (see fig. 4.20) are derived from another type of non-overlapping generator shown in fig. 4.31. Two cross-coupled NOR gates with a couple of inverters in their outputs are used for generation of the non-overlap. These gates are constructed using larger than the minimum length transistors to obtain a delay corresponding to the required non-overlap. The latter should ideally be no more than one or two ns. F_{NO} is another internal pulse whose frequency is the same as f_1 and f_2.

Simulation results indicate that for f_D = 1MHz, the first version of the complete clock generator circuit dissipates an average power of 438μW, whereas the second version dissipates 370μW. Note that for the above decoding rate, the fastest master clock f_{MS2} operates at 64MHz.

4.3.9 Simulated Performance

The three different versions of the MFD will be denoted here as MFD_1, MFD_2 and MFD_3. They are all constructed using the corresponding blocks and circuits described in the previous sections. In particular, in MFD_1 there is no clock generator circuit and all the clock signals are generated externally at the expense of sixteen extra pads. Also, in MFD_3 there is no front-end circuitry as more freedom on the experimental testing can be obtained including software driven testing procedure, common in the


(a)


(b)

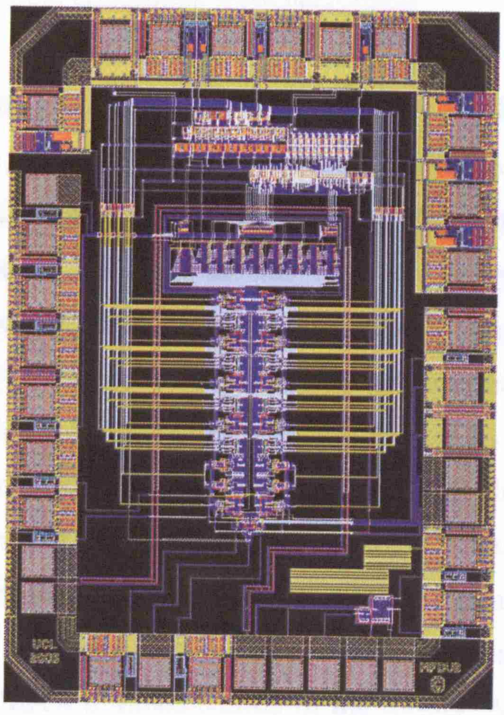

(c)

Fig. 4.32 The complete layout of: (a) MFD$_1$, (b) MFD$_2$, (c) MFD$_3$.

literature [109], [110] and [112]. The layout of each version of the MFD is shown in fig. 4.32. Their post-layout simulated performance and other characteristics are also given in table 4.14. The decoders were simulated for various pseudo random binary input sequences of up to 1024 bits length. These sequences were inserted in the encoder of fig. 2.6, which was directly connected to the MFD. This allowed the

Table 4.14 Simulated* performance and characteristics for the MFDs.

Parameter	*MFD₁*	*MFD₂*	*MFD₃*
Operator	MAX	MIN	MIN
AMS technology	CMOS 0.8μm	CMOS 0.6μm	CMOS 0.6μm
Core size (mm^2)	1	0.5	1.45
Maximum speed (Mb/s)	0.84	1.25	1
Average power (mW)	3.9	4	2.45
Bit metric (μA)	1	1-2	1-2
Analogue core range (μA)	10-30	10-30	5-40

*Post-layout simulations

simulation of the complete encoder-decoder circuit under an ideal, noiseless channel. Since there is no simple way of verifying the BER performance of the decoders using CAD tools, extensive simulations under only ideal conditions were performed.

Comparison of the results of table 4.14 with preliminary simulation results of the complete decoders indicate that the parasitics introduced by a weak layout can have a significant effect on performance. This effect is pronounced more on MFD$_1$ where preliminary simulations verified a data rate of 2.4Mb/s. In this case the layout degrades speed by approximately three times. Various analogue and mixed-signal layout techniques [137] have been adopted in the following versions of the chip resulting in minor speed degradations. Note that the data rate f_D is determined by $T_D = 8 \cdot T_S$, where $T_S = T_{ACS} + T_{BMC}$. Also, the clock generator is designed such that the delay imposed by the WTA block, T_{WTA}, cannot exceed T_S (i.e. $T_{WTA} = T_S$). This means that even if T_{ACS} and T_{BMC} are sufficient for their purpose, T_S and hence T_D can be inadequate and limited by the WTA. Also, the power dissipation of MFD$_1$ is higher than the other versions and this is due to the MAX implementation of the former. As discussed in section 3.5.2, the required analogue core (ACS and WTA) range for a MAX MFD is at least two times higher than that of a MIN MFD. Although these results were for a soft-decision MFD, modelling and simulation of a hard-decision MFD in Simulink® reveals similar figures.

4.3.10 Experimental Verification of Fabricated Chips

All versions of the MFD described in the previous sections were fabricated and tested in the laboratory. MFD_1 was very difficult to test due to the lack of an internal clock generator. Initially the clocking circuitry was assembled on a bread board using discrete components. Although the speed of most commercial standard digital components was satisfactory and sufficient for the MFD experimental verification, specifications such as rise and fall times limited timing control, which in turn made synchronisation almost impossible. Exact synchronisation is very important in this case, as it determines the validity of BER measurements. As an alternative, a Xilinx® FPGA board was programmed to the required clock specifications. Better accuracy was achieved this way with entire control over the clock signals. However, interfacing the FPGA with the MFD imposed additional limitations, resulting in non-consistent measurements and ambiguous results.

In MFD_2 the above concerns are eliminated since a clock generator block is included on the chip (see section 4.3.8). Also, in order to verify the functionality of the ACS circuit, an additional ACSU (ACS circuit 2 of fig. 4.15 and SI cell of fig. 4.19) is incorporated as a test structure. Due to a number of reasons, MFD_2 failed to provide reliable BER measurements. Comparison of the test structure with post-layout simulations revealed that, in practice, the RCC of fig. 4.15 has approximately four times lower resolution. This means that the bit metric value cannot be smaller than $4\mu A$. Although the analogue core input range is variable in the MFD_2 (see table 4.14), it is incapable of accommodating such a large scale. The minimum required range in a hard-decision MFD is 3.5 equivalent-bits, according to Simulink® simulations. This translates to an upper limit of $55\mu A$, which is beyond the analogue core range. The poor RCC resolution was addressed in ACS circuit 3, as discussed in section 4.3.5.1. Also, another important factor that limits the performance of MFD_2 is a misconnection in the feedback network. An extra DFF was connected in series with the other flip-flops and essentially delayed the feedback output by another decoding cycle. This is considered as a system wiring mistake that cannot be verified by ideal simulations (i.e. assuming transmission under a noiseless channel). Since the decoder is error correcting, under a noiseless channel, it can potentially correct errors arising from various misconnections. This is also verified using system-level modelling. The simulation of a complete MFD in Cadence® is very time and space consuming and this limits the size of an investigation. In the first two versions of the MFD, emphasis was mostly given in the output of the decoder, and compared with the input sequence it determined a decision as to whether or not it works.

The above helped to develop a working MFD3 by resolving circuit issues and misconnections including on-system test structures. This was achieved by monitoring the current values of ACS and WTA outputs and comparing them with system-level simulations for certain input sequences. MFD3 also features improved matching, as discussed in previous sections, to ensure good performance and functionality.

A detailed description of the MFD3 testing procedure follows next. Experimental performance characteristics and various BER measurements are given in section 4.3.10.2.

4.3.10.1 Test Setup

Fig. 4.33 shows the setup diagram for the BER measurements of the MFD. The sequence **s** is generated in an arbitrary waveform generator and represents the encoded data for various binary pseudo random input sequences **u** of up to 1024 bits long. It is already converted into antipodal signals of amplitude A, this way realising the required modulation. An AWGN generator of variable average power N adds

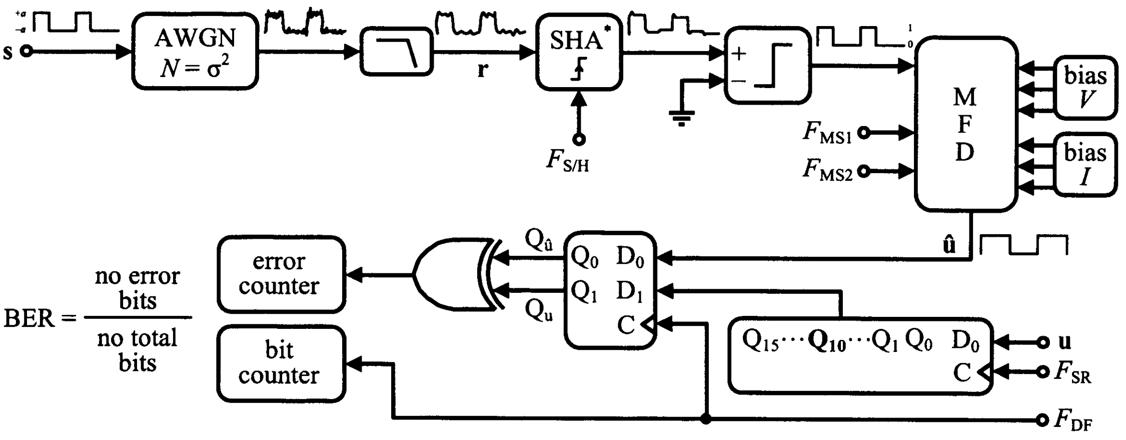


Fig. 4.33 BER measurement setup (*where SHA is the sample and hold amplifier).

noise onto the modulated signal. The noisy waveform is then passed through a *low pass filter* (LPF) resulting in the received sequence **r**. Since MFD$_3$ features no front-end circuit, **r** needs to be sampled and then converted into binary as discussed in section 4.3.1. This is realised using a commercial *sample and hold amplifier* and a comparator. The output of the comparator corresponds to the received encoded data which interfaces directly with the MFD chip. The bias voltages for the decoder are generated externally using a voltage reference and potential dividers (i.e. resistors). V_{B1}, V_{B2} and V_{B3} are 1V, 1.5V and 1.75V, respectively. Also, the bias currents I_B, I_{BP}, and I_{BS} are supplied by three current sources controlled by variable resistors (i.e. trimmers). I_B can vary from 0μA to 10μA, whereas I_{BP} and I_{BS}, which define the parallel and sequential bit metric values, can vary from 4μA to 8μA. The decoder outputs the estimated sequence $\hat{\mathbf{u}}$ that needs to be compared with the actual information sequence **u**. In order to do so, a shift register is used to introduce a delay on **u** and synchronise the two sequences. The delay can be adjusted between integer numbers of decoding cycles with the aid of a manual switch. Further fine tuning can be obtained by the external clocks $F_{S/H}$ and F_{SR}. This delay is about eleven decoding cycles T_D, accounting for $10 \cdot T_D$ for the SS to contain useful information (i.e. twenty code bits corresponding to the first sliding block) and about one T_D for the SS to perform the demultiplexing (see fig 4.5) and for the front-end setup processing of fig. 4.33. The final synchronisation stage for **u** and $\hat{\mathbf{u}}$ consists of two DFFs (in one package) clocked by the same F_{DF}. The outputs Q_u and $Q_{\hat{u}}$ should be *non return to zero* (NRZ) signals, necessary for the counter* to detect the difference between the two sequences. This is achieved by resetting the former outputs using the inverse of clock F_{DF} (not shown in fig. 4.33). The XOR function between Q_u and $Q_{\hat{u}}$ should then generate a logical "1" for every different bit in the sequences. In other words, a falling edge is generated for every bit in error. Finally, two counters are used to produce the bits in error and the total number of bits, a division of which gives the BER.

All clock signals **s** and **u** are generated from four different generators in the laboratory (see table 4.15). $F_{S/H}$ is twice as fast as F_{SR} and F_{DF}. The latter operate at the decoding rate f_D. BER measurements for various SNR values were taken in steps of 1dB. These values are converted into E_b/N_o in order to plot the typical BER against bit energy per

* The counter detects falling edges. A NRZ binary sequence has a falling edge for every bit.

noise power spectral density curves. This is done as follows. The energy per information bit E_b is twice the energy per code symbol E_s since the employed convolutional code is a rate $R = 1/2$. E_s is equal to the antipodal signal power S times its bit time T_s [3]. Assuming the power is normalised to a resistance of 1Ω, E_b can be expressed as

$$E_b = 2 \cdot E_s = 2 \cdot S \cdot T_s = 2 \cdot A^2 \cdot T_s \tag{4.6}$$

The power spectral density N_0 is the noise power N divided by the bandwidth W [3]. Assuming the same normalisation as above, N_0 can be expressed as

$$N_0 = N/W = \sigma^2/W \tag{4.7}$$

where σ^2 is the variance of the output noise. From (4.6) and (4.7) E_b/N_o is

$$E_b/N_o = 2 \cdot A^2 \cdot T_s \cdot W/N \tag{4.8}$$

A can be set to any value in the waveform generator, T_s is the bit time that determines the decoding speed (i.e. $T_D = 2 \cdot T_s$) and W is the bandwidth of the LPF. Equation (4.8) can be expressed in dB as

$$(E_b/N_o)_{dB} = 10 \cdot \log (2 \cdot A^2 \cdot T_s \cdot W/N) \tag{4.9}$$

Hence, by varying the noise power, nine sets of E_b/N_o and BER values are obtained for each test.

The complete test setup as constructed in the laboratory is shown in fig. 4.34. Also, details of all the devices used are given in table 4.15. Three small sized *printed circuit*

Table 4.15 List of all devices used in fig. 4.34.

Manufacturer	Description	Model	Purpose
Agilent	Digital Oscilloscope	Infinium 1GHz	probing
Hewlett Packard	Pulse/Pattern Generator	81110A	clocks/sequences
Hewlett Packard	Pulse Generator	8082A	clocks
Noise/Com	Noise Generator	UFX 7108	AWGN
TT*i*	Triple Power Supply	PL320QMT	power supplies
Agilent	Triple Power Supply	E3631A	power supplies
Hewlett Packard	Universal Counter	53131A	counting

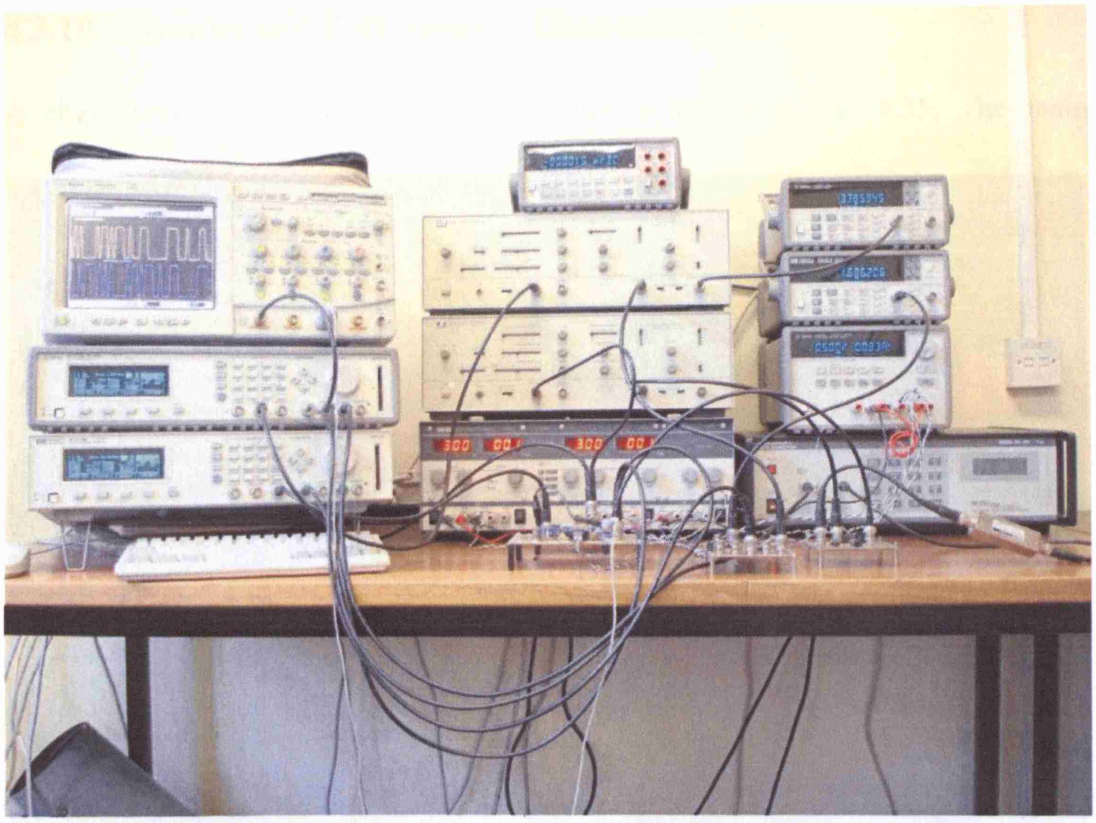


Fig. 4.34 Laboratory setup for prototype BER measurements.

boards (PCB) were built in order to accommodate testing of the MFD. The main PCB contains the prototype chip including the relevant voltage references and current sources. The sample and hold amplifier, the comparator and the remaining digital components shown in fig. 4.33 were constructed on another PCB using separate supplies. Also, a PCB was used to implement a clock divider circuit for synchronisation of the pulse and the pattern generators. A list of the main components used is shown in table 4.16.

Table 4.16 List of main components used for the construction of the test PCBs.

Manufacturer	Description	Model	Purpose
Analog Devices	Voltage Reference	REF191GP	bias voltages
National Instruments	Current Source	LM334Z	bias currents
Analog Devices	Sample and Hold	AD783JQ	front-end sampling
Analog Devices	Comparator	CMP 402	threshold detector
Philips	Dual DFF	74F74N	clock divider
Philips	Hex DFF	74HC174	shift register
Philips	Quad 2-In XOR	74HC174	XOR

4.3.10.2 Results and Performance Characteristics

A chip micrograph of the complete decoder is shown in fig. 4.35. The main

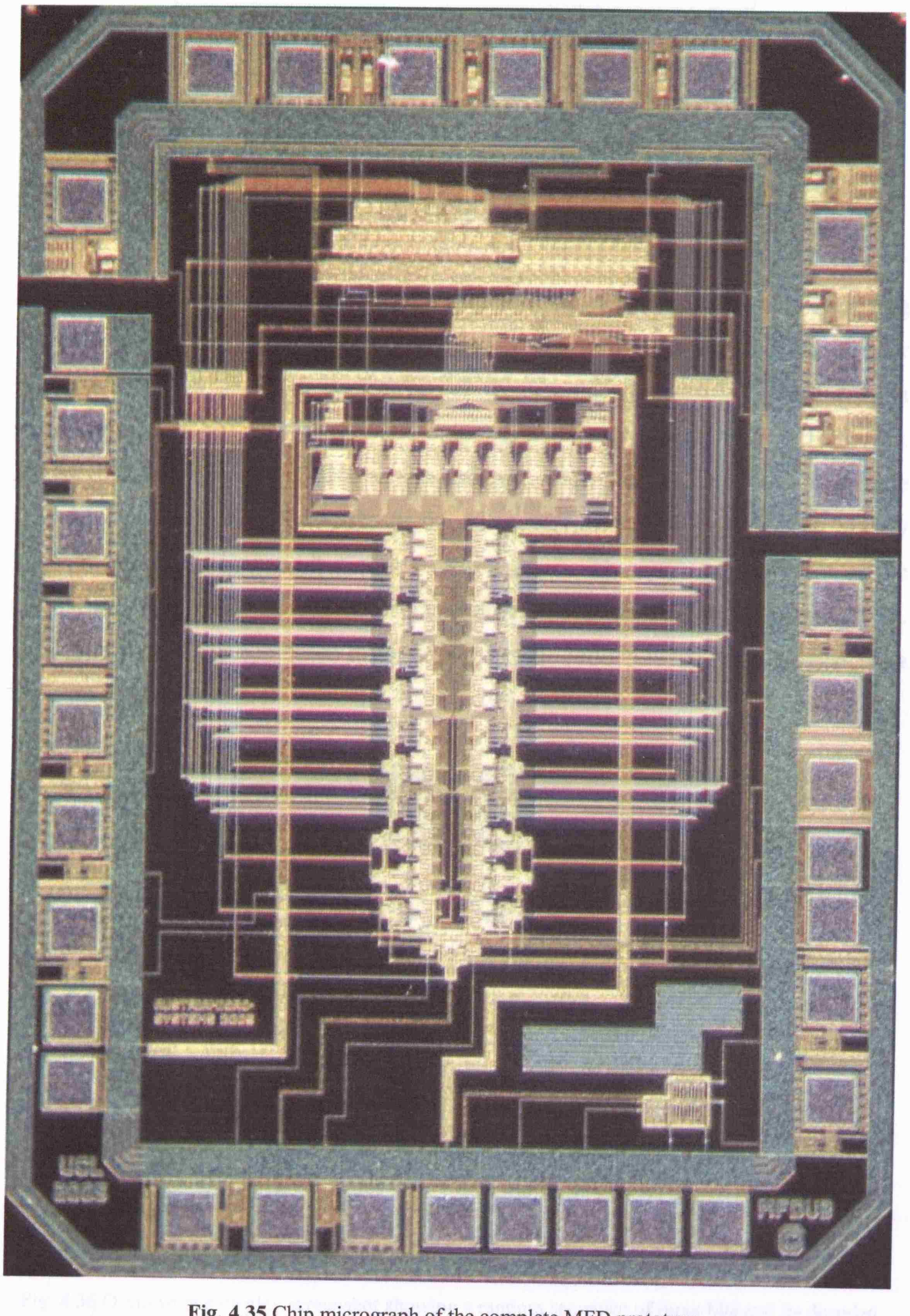


Fig. 4.35 Chip micrograph of the complete MFD prototype.

Table 4.17 Main characteristics of the fabricated MFD.

Technology	AMS CMOS 0.6μm 3M2P
Supply voltage (V)	3
Core area (mm^2)	1.45
Chip area (mm^2)	3.54
Power dissipation (mW) @ $I_B = 5\mu A$, $I_{BS} = I_{BP} = 5\mu A$	2.45
Data rate (Mb/s)	1
CG (dB) @ BER = 10^{-5}	1.45

characteristics are summarised in table 4.17. The total chip area is 3.54mm^2 including the pads and a test structure circuit located in the bottom right used for another project. Fig. 4.36 shows a screenshot of the oscilloscope display, illustrating a sequence of data bits and its decoded output (delayed by eleven bits) at 1.25Mb/s. Although the decoder can operate at that data rate, BER measurements are acceptable only for data rates of up to 1Mb/s. At the latter rate, the power consumption is only 2.45mW and the required bit metric step is 1.25μA. The BER performance of the prototype MFD operating at 1Mb/s, for various bit metric values, is shown in fig. 4.37. The loss in CG compared to system-level simulations is less than 0.15dB for a bit metric step of 1.25μA.

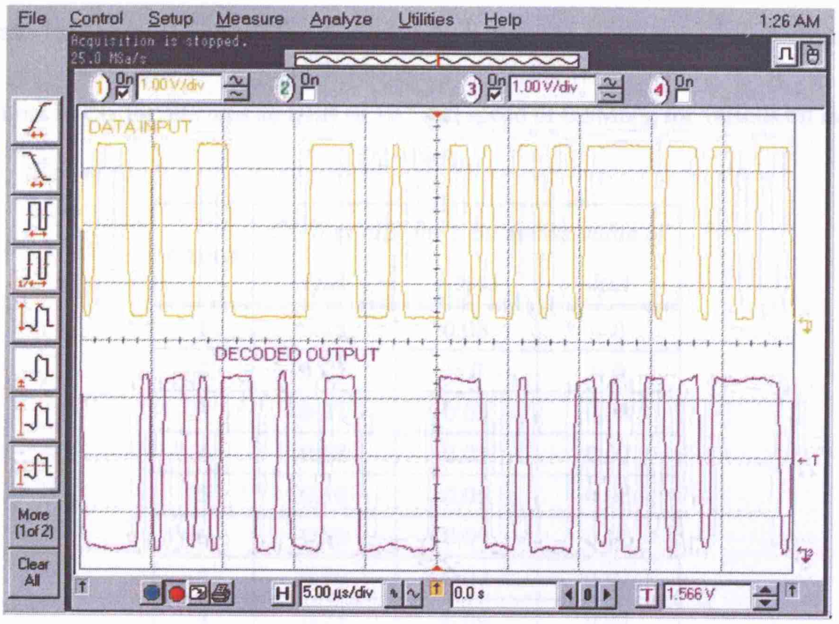

Acquisition is stopped.
25.0 MSa/s
1 On 1.00 V/div
3 On 1.00 V/div
H 5.00 μs/div
0.0 s
T 1.566 V

Fig. 4.36 Oscilloscope display screenshot showing a random sequence of input bits and its decoded output at 1.25Mb/s.

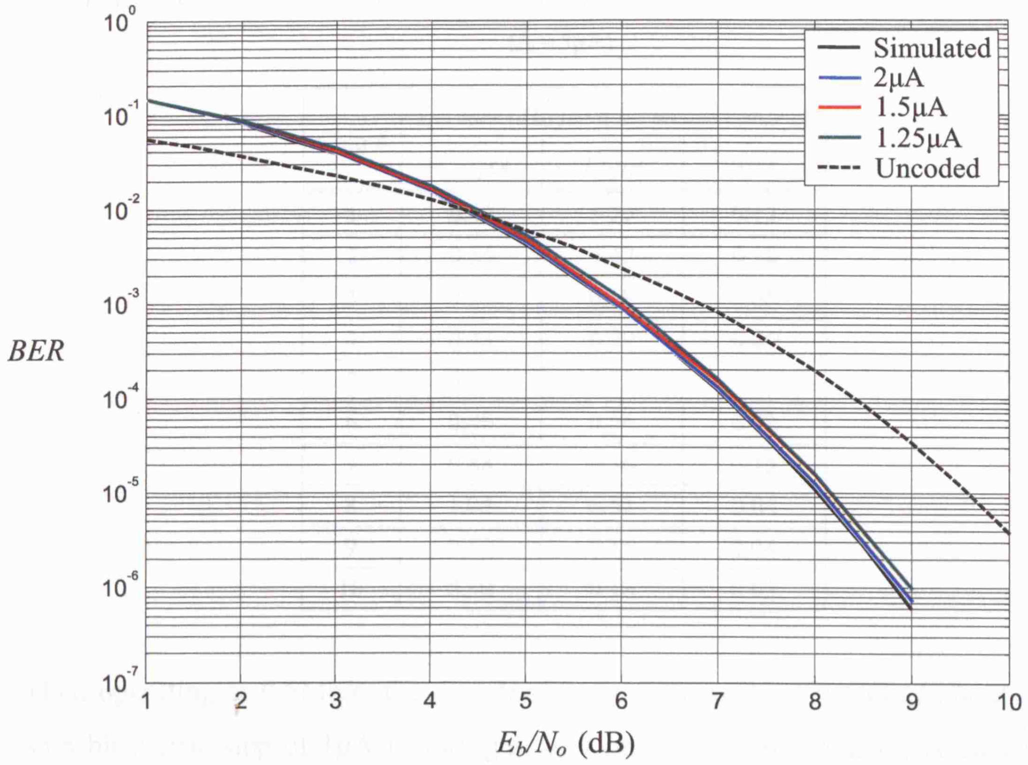


Fig. 4.37 BER measurements @ 1Mb/s for various bit metric steps and $I_B = 5\mu A$.

Ten prototype chips were measured in total resulting in 100% yield. As expected, a slight variation in performance characteristics from chip to chip was observed. Tables 4.18 and 4.19 summarise the loss in CG at a BER of 10^{-5} for data rates of 0.5Mb/s and 1Mb/s respectively, for all chips and three different bit metric values. Note that for any data rate less than 0.5Mb/s the decoder can operate with zero initial bias I_B. Also,

Table 4.18 Loss in CG for all chips @ BER of 10^{-5} and speed of 0.5Mb/s, for various bit metric values ($I_B = 0\mu A$).

Chip #	*CG loss (dB) for a bit metric value of*		
	1μA	*1.5μA*	*2μA*
1	0.28	0.08	≈ 0
2	0.27	≈ 0	≈ 0
3	0.31	0.03	0.08
4	0.18	0.01	0.07
5	0.18	0.05	0.06
6	0.20	0.09	0.12
7	0.30	0.07	0.02
8	0.21	0.09	0.07
9	0.25	0.08	0.07
10	0.23	0.10	0.04

Table 4.19 Loss in CG for all chips @ BER of 10^{-5} and speed of 1Mb/s, for various bit metric values ($I_B = 5\mu A$).

Chip #	*CG loss (dB) for a bit metric value of*		
	1.25μA	*1.5μA*	*2μA*
1	0.42	0.20	0.12
2	0.85	0.42	0.18
3	0.29	0.18	0.08
4	0.48	0.29	0.06
5	0.21	0.12	0.09
6	0.46	0.16	0.02
7	0.88	0.36	0.15
8	1.05	0.52	0.04
9	0.13	0.11	0.04
10	0.30	0.18	0.05

when operating at 0.5Mb/s, the decoder has an average loss in CG of about 0.24dB with bit metric step of 1μA featuring a very low power of 1.55mW. A very similar average loss in CG is achieved at 1Mb/s when the bit metric is set to 1.5μA. The average power consumption then is 2.6mW. The maximum deviations from the average of the measured results of tables 4.18 and 4.19 are shown in tables 4.20 and 4.21, respectively. Repetition of several experimental measurements indicates that the maximum measurement error is about 0.05dB. This means that any deviation of less than ± 0.05dB can potentially be considered as negligible.

Table 4.20 Maximum deviation from the average of the results of table 4.18

Bit metric (μA)	Average loss in CG (dB)	Maximum deviation (dB)	
		+	–
1	0.241	0.069	0.061
1.5	0.060	0.040	0.060
2	0.053	0.067	0.053

Table 4.21 Maximum deviation from the average of the results of table 4.19

Bit metric (μA)	Average loss in CG (dB)	Maximum deviation (dB)	
		+	–
1.25	0.507	0.543	0.377
1.5	0.254	0.266	0.144
2	0.083	0.097	0.063

Assuming that a maximum of 0.1dB is an acceptable loss in CG and taking into account the performance spread over the ten prototypes, the decoder can operate at a data rate of 0.5Mb/s with a bit metric current of 1.5μA featuring a yield of 100%. Also, for a 0.15dB acceptable loss, the decoder operates at 1Mb/s with a bit metric of 2μA resulting in 90% yield. The energy efficiency is 3.8nJ/b and 2.95nJ/b for the former and latter cases, respectively. A better efficiency can be achieved providing a higher loss in CG can be tolerated. Clearly, the energy efficiency is not linearly dependent on the speed of operation and higher speed does not necessarily mean higher energy consumption. This is closely related to the fact that although the digital power increases with frequency*, the analogue circuits used in the MFD can benefit in terms of power consumption from higher data rates (see section 4.3.5.1). This is mainly due to the fact that the clock generator circuit used is not optimised for low power. The aim of the clock generator in the MFD is to achieve synchronisation for all the blocks that scales with frequency, with the lowest possible circuit complexity.

4.3.11 Comparison with Existing Analogue Decoders

Although the hard-decision MFD described here serves as a first time realisation of the MFDA principle, a useful comparison of performance characteristics with existing analogue decoders should be made. In order to perform a thorough and fair comparison, characteristics such as speed and power should be combined with the loss of CG from system-level simulations resulting in normalised features. The final remark then would be the fact that a soft-decision MFD benefits from a 2dB gain over a hard-decision version and can be constructed from simple modifications to the latter decoder without major performance degradation.

The energy efficiency per decoded bit is a common measure of comparison of analogue decoders. However, many of these decoders have very low yield and/or suffer large CG losses. Therefore, a suitable normalisation for the energy efficiency must be adopted in order to accommodate possible CG degradation. This can be achieved by multiplying the energy efficiency by the ratio of the ideal CG over the

* the power consumption in digital circuits is proportional to $f \cdot C \cdot V_{DD}^2$

measured CG, comparing effectively the energy efficiency per decoded bit for a certain CG loss. The value of this normalisation can be demonstrated with the aid of an example. If two decoders A and B feature the same energy efficiency and ideal CG, and A incurs a lower measured CG (i.e. higher CG loss) than B, then a coefficient equal to the ratio of their measured CG will favour the energy efficiency of decoder B. This of course results in a hypothetical measure of energy efficiency (that will here be referred to as *coding performance normalised energy efficiency* that only applies in practice if the performance of A can be improved in order to match that of B, at the cost of increased energy efficiency which is proportional to the ratio of their measured CG. Although the latter cannot always be guaranteed, the coding performance normalised energy efficiency still provides a useful measure of comparison between decoders of different applications and coding schemes. In the following comparisons the CG values are taken at a BER of 10^{-5}.

Table 4.22 shows a summary of performance characteristics including the coding performance normalised energy efficiency for all analogue decoders mentioned in section 4.2. These results are based on measurements of prototype chips and so the CG is not always available. The coding performance normalised energy efficiency is therefore calculated only for decoders with measured BER results. It is very difficult

Table 4.22 Measured performance characteristics comparisons combining tables 4.1 and 4.2.

Ref.	Data rate (Mb/s)	Power (mW)	Energy (nJ/b)	CPNEE* (nJ/b)	Area (mm^2)	Comments/Description
[8]	50	89	1.78	3.09	3.24	2 dicodes in parallel
[9]	200	30	0.15	0.22	0.5	2 dicodes in parallel
[11]	115	39	0.34	0.39	1	Complete VD
[12]	200	55	0.275	N/A	0.78	SSM not included
MFD	*1*	*2.45*	*2.45*	*2.67*	*1.45*	*Hard-decision*
[119]	100	50	0.5	N/A	1.19	Tail-biting BCJR
[120]	320	20	0.0625	0.068	0.12	Tail-biting BCJR
[110]	1	45.2	45.2	52.7	0.81	Tail-biting BCJR
[109]	13.3	185	13.9	21.18	1.32	Turbo
[112]	2	7.6	3.8	4.3	4.07	UMTS Turbo

*Where CPNEE is the coding performance normalised energy efficiency.

to perform a fair comparison in all aspects of the decoders of table 4.22 since they implement different algorithms and coding schemes and are also aimed at different applications. For example, in terms of power dissipation and for a certain application that requires a data rate of 1Mb/s, the MFD seems to outperform all other decoders. Although this may be valid, since the analogue power is not always proportional to the speed of operation, the latter statement cannot be shown. So, in fact, there is only decoder [110] which the MFD can be strictly compared with since there are measurements at 1Mb/s. Therefore, the comparison can only be limited to the extent of showing table 4.22, including all available parameters for all decoders, from where individual features can be extracted.

As a final remark, the convolutional decoders shown in table 4.22 are all mixed-signal since they implement the VA. The only entirely analogue convolutional decoder can potentially be a soft-decision MFD that can be easily constructed using the circuits described in this chapter.

4.4 Implementation Considerations for a Soft-Decision MFD

A soft-decision MFD mainly differs from a hard-decision version in the following two aspects: the SS and the BMC structure. As it was described in section 3.5.2, the input to a soft-decision MFD is analogue and therefore the SS consists of an analogue delay line capable of storing $2L$ sampled values. This means that there is no need for a front-end comparator and the SS block stores analogue sampled values of an input range of 3-equivalent bits. Since the contents of the SS are no longer digital signals the BMC cannot consist of simple current sources. Suitable transconductors that convert the analogue voltage samples into currents should be adopted to accommodate the required linear range. These can be realised using differential pairs with multiple current outputs, which can be easily designed to achieve high linearity within a 3-equivalent bit range.

4.4.1 SS Analogue Delay Line

Figure 4.38 shows a circuit diagram for the analogue SS block used in any R = 1/2 MFD. As it was described in section 3.4.1, the SS consists of two identical delay lines

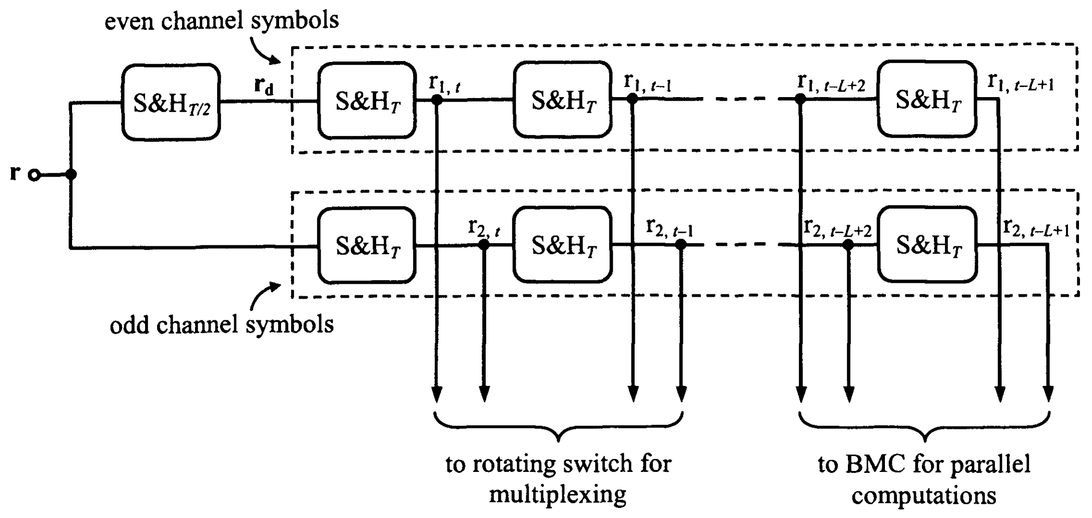


Fig. 4.38 Circuit diagram of a soft-decision MFD SS consisting of two analogue delay lines of length L. $S\&H_T$ denotes a sample and hold circuit clocked at T.

one for each channel symbol. Each delay line consists of L cascaded sample and hold circuits clocked at the data rate T. In order to demultiplex the input **r**, as it was described in section 4.3.2 (see fig. 4.5), an extra sample and hold circuit clocked at twice the data rate (i.e. $T/2$) is used at the front of the upper delay line.

There are two issues associated with the direct implementation of the delay lines shown in fig. 4.38. A simple sample and hold circuit consists of a switch, a capacitor and a unity gain buffer. The dynamic operation of such a circuit requires the use of two sample and hold circuits in series for each stage to acquire a new input while storing the previous sample for one clock period T. Therefore, the use of $2L$ capacitors and $2L$ buffers is necessary to implement each delay line. This raises size and complexity issues and most importantly accuracy issues. Although Simulink® simulations show that low accuracy sample and hold stages can be used (see section 3.5.3.3), cascading $2L$ non ideal unity gain buffers can result in significant loss of information. A better approach can be followed by implementing a parallel delay line and this is shown in fig. 4.39 [138]. This is an L-tap delay line where only $L + 1$ sample and hold circuits are used at the cost of an extra $L + 1$ rotating switches. This structure improves performance and power dissipation since the input signal suffers only the distortion of one sample and hold circuit and the number of unity gain buffers is reduced by $L - 1$. As the timing diagram of fig. 4.39 illustrates, the $L + 1$ sample and hold circuits work in a manner such that at each period T a new value is acquired by one of C_0-C_L while L previous sampled values are held by the remaining

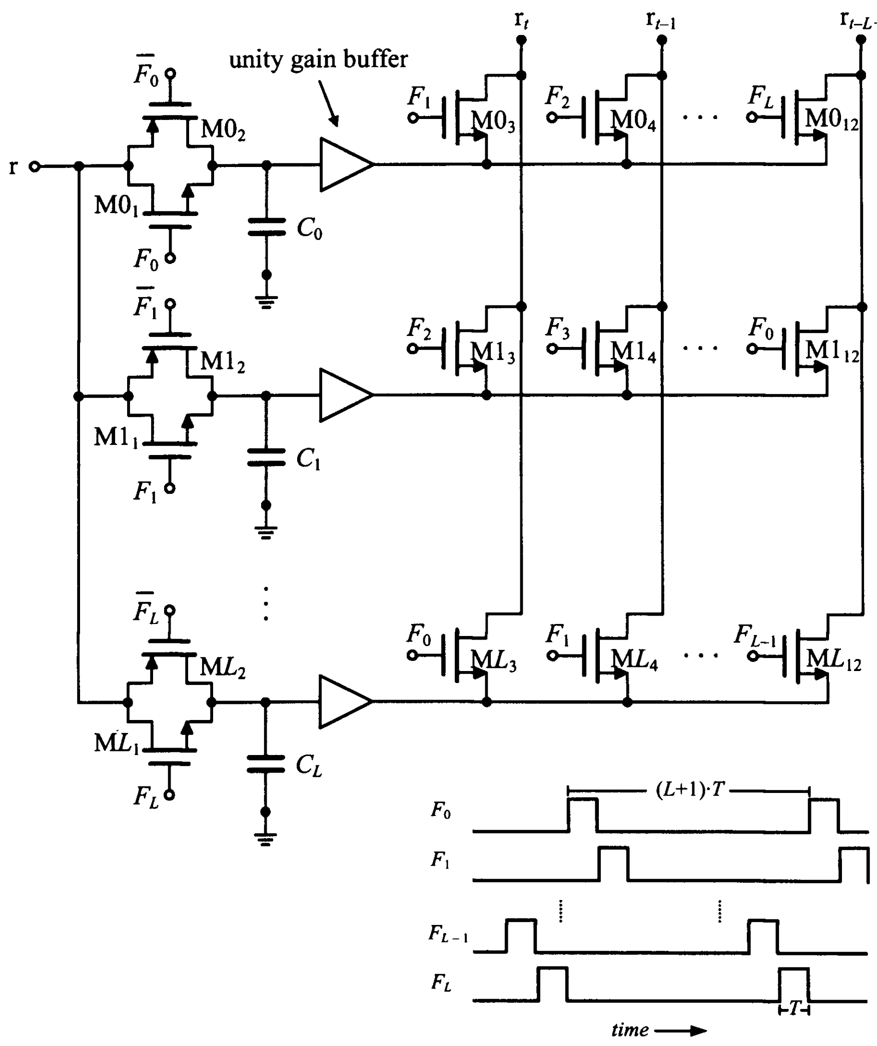


Fig. 4.39 Parallel implementation of the delay line of fig. 4.38 and timing diagram.

capacitors. The key point in this design configuration for low complexity and low power operation is the choice of a suitable unity gain buffer circuit. A simple source follower could be used instead of a single stage differential amplifier in order to improve bandwidth and bias current requirements. However, the former suffers from large DC voltage drops from input to output (i.e. V_{GS}) which can be signal-level dependent. This appears as an offset at the output and requires extra circuitry to be compensated for, which defeats the object of simple buffer architecture.

An excellent candidate in this case is the circuit shown in fig. 4.40. This is in fact an inverting PMOS common source amplifier with an identical diode connected transistor load and unity gain [139]. The advantage of using PMOS devices in CMOS processes that incorporate twin well fabrication is the fact that their body can be connected to any potential. In this case the body of M_1 is tied to its source eliminating

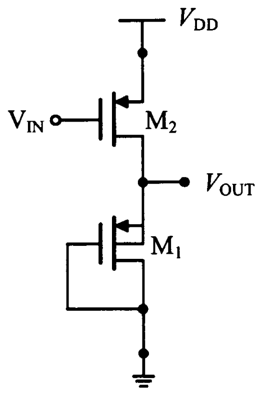


Fig. 4.40 A simple PMOS common source amplifier with negative unity gain.

the *body effect* (i.e. threshold voltage V_{T1} dependency on V_{OUT}). This in turn improves matching of M_1 and M_2 and hence linearity. Since the same current flows through M_1 and M_2, they exhibit the same V_{GS}. Therefore, for long channel devices where the *channel length modulation* can be ignored, large signal analysis results in $V_{OUT} = V_{DD} - V_{IN}$. This can simplify to $V_{OUT} = V_{DD}/2$ for a DC level input of $V_{IN} = V_{DD}/2$. On the other hand, small signal analysis gives:

$$A_v = -\frac{g_{m1}}{g_{m2} + g_{ds2} + g_{ds1}} \tag{4.10}$$

where A_v is the small-signal voltage gain, g_{m1}, g_{m2}, g_{ds1} and g_{ds2} are the transconductances and conductances of M_1 and M_2, respectively. And since usually $g_{m1} >> g_{ds1} + g_{ds2}$

$$A_v \approx -\frac{g_{m1}}{g_{m2}} \approx -1 \tag{4.11}$$

The inverter of fig. 4.40 can operate easily at 1MHz drawing a current of about 5μA. In fact, simulations show that for V_{DD} = 3V and a peak to peak voltage amplitude of V_{IN} = 200mV centred around 1.5V the gain of the circuit is $-1.0012 < A_v < -0.999$.

The complete SS circuit for a soft-decision version of the MFD described in this chapter has been designed and simulated using AMS CMOS 0.6μm technology. The circuit is based on the diagram of fig. 4.38 using two identical 10-tap delay lines of the type shown in fig. 4.39. The unity gain inverter of fig. 4.40 is used as a buffer. The inversion in the lower delay line is compensated for by adding an extra inverter at the

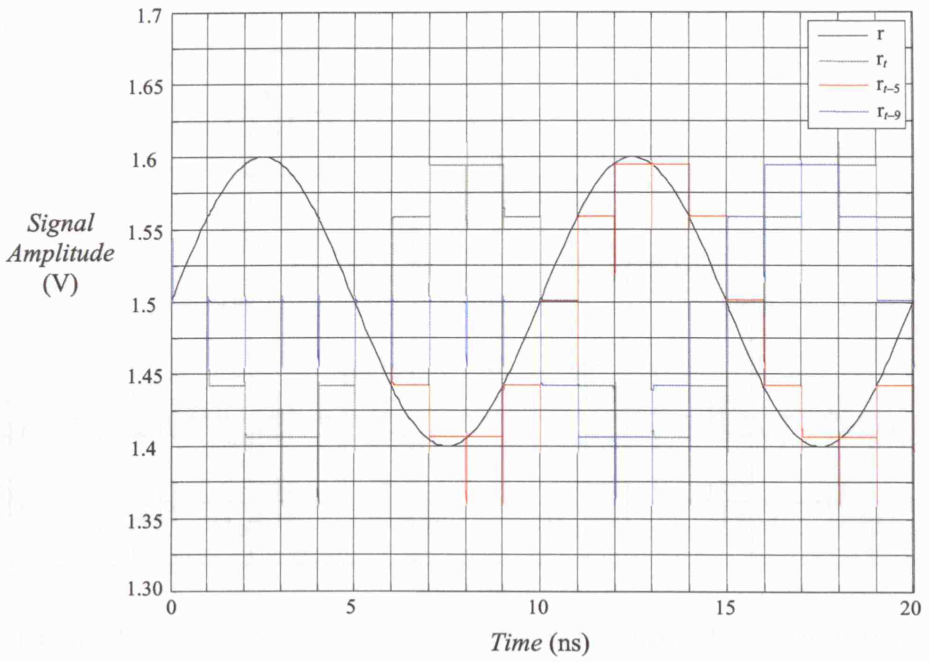


Fig. 4.41 Simulation results for a 10-tap version of the delay line of fig. 4.40.

front. This is not necessary in the upper delay line since it already uses an extra sample and hold stage that operates at the channel rate (i.e. S&$H_{T/2}$ in fig. 4.38). The aspect ratio of all CMOS switches in the delay lines is W/L = 3/0.6 μm and that of the remaining NMOS switches is W/L = 1.4/0.6 μm. Also, the aspect ratio of the identical transistors in the unity gain inverters of the delay lines is W/L = 2/2 μm. However that of the unity gain inverter used at the front of the upper delay line is W/L = 6/2 μm in order to operate at twice the bandwidth. The latter inverter draws a current of 18μA and together with the 22 inverters of the delay lines they constitute a total of 128μA. The complete SS dissipates an average power of about 0.52mW when operating at a data rate of T = 1MHz. Fig. 4.41 shows simulation results for one of the 10-tap delay lines. A sine wave as an input and three tap outputs are shown in the figure for illustration purposes. Note that the outputs are delayed and inverted sampled versions of the input.

4.4.2 BMC Transconductors

The analogue samples from the SS need to be converted to equivalent current values in order to interface with the ACS and WTA circuits described in section 4.3. This is done in the BMC using PMOS differential pairs with independent current mirror

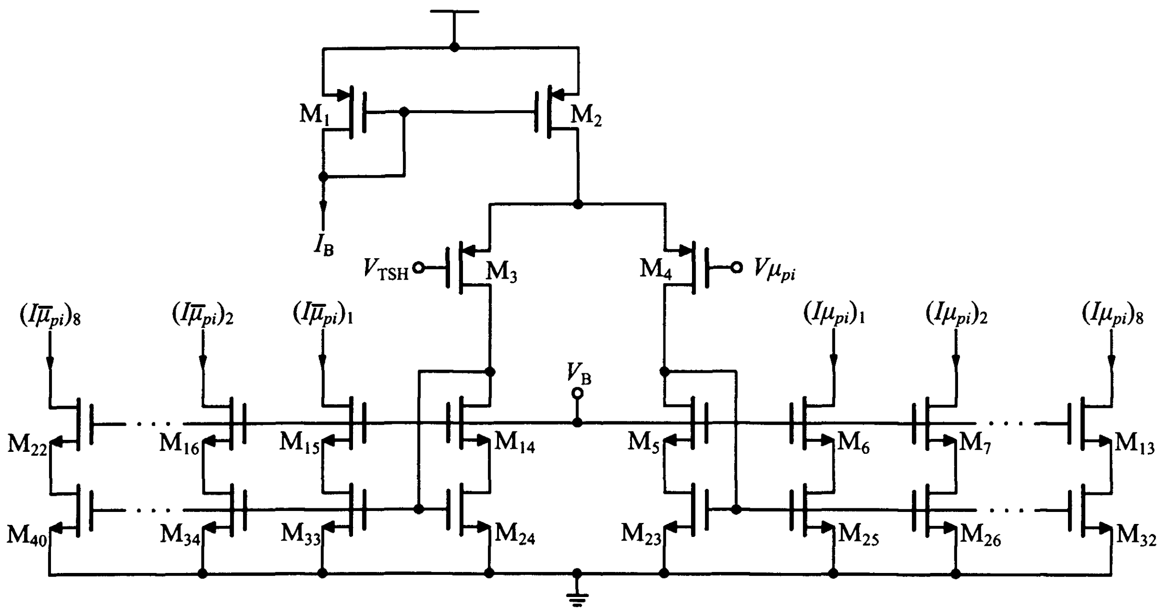


Fig. 4.42 Fixed-state symbol metric generator circuit used in a soft-decision MFD BMC.

loading. Fig. 4.42 shows a BMC transconductor responsible for the generation of the fixed-state symbol metrics on a modified tree (see fig. 3.5). Four transconductors of the type shown in fig. 4.42 are used for the generation of all fixed-state branch metrics; two for the sequential and two for the parallel computations. The input voltage $V\mu_{pi}$ is generated in a similar manner as in the BMC of fig. 4.7. The only difference here and main advantage of the soft-decision BMC is that the complimentary voltages shown in fig. 4.7 are generated by the transconductors without the use of digital circuitry. One side of the transconductor (i.e. eight $I\mu_{pi}$ current outputs) provides the symbol metrics corresponding to a code bit of "1" on the modified tree, while the other side provides the symbol metrics corresponding to a "0". The need for eight replications of each symbol metric is explained as follows. A closer observation of the modified tree of fig. 3.6 reveals that at any time instant after

Table 4.23 Transistor dimensions for the circuit of fig. 4.42.

Transistor	W (μm)	L (μm)
M_1-M_2	20	1.5
M_3-M_4	12	2
M_5-M_{22}	4	0.6
M_{23}-M_{24}	4	2
M_{25}-M_{40}	8	2

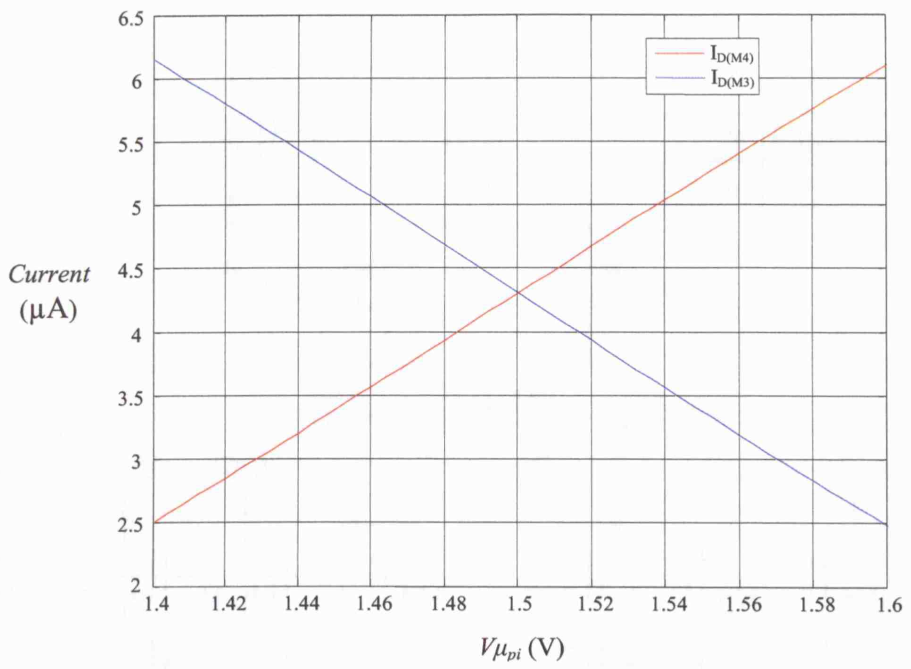
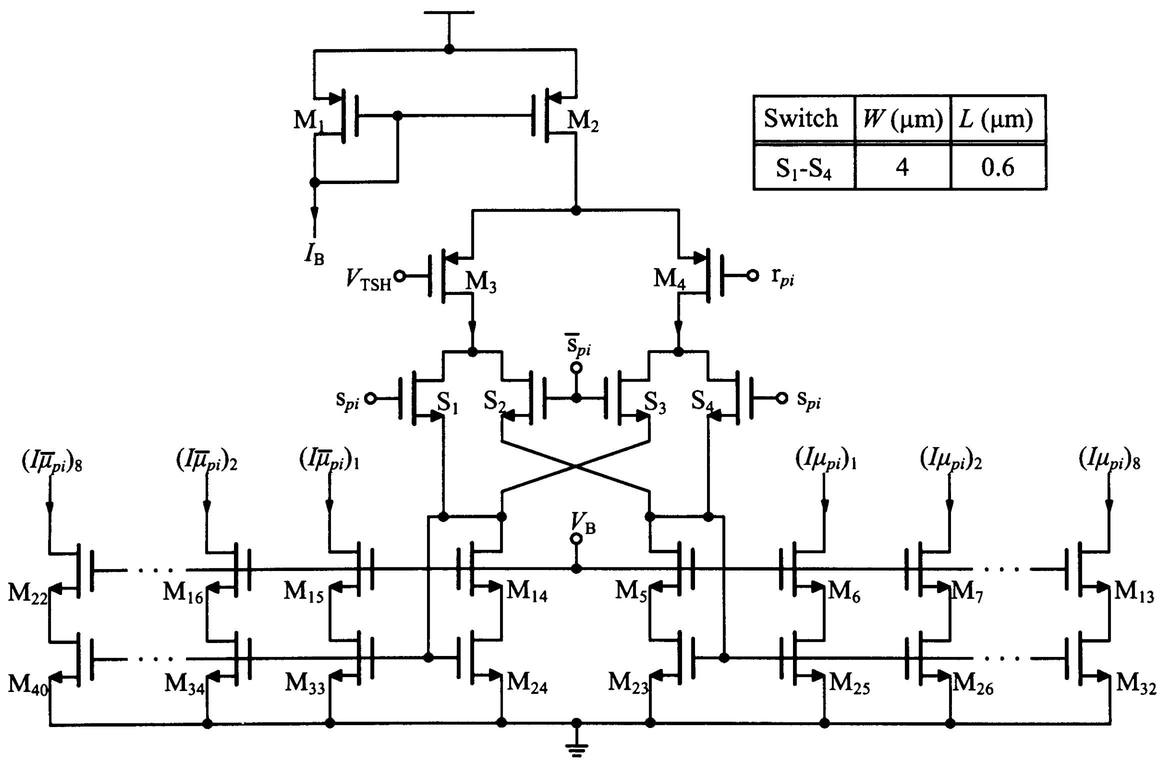


Fig. 4.43 DC response of the transconductor shown in fig. 4.42.

t_1, there are eight "0" and eight "1" code bits in all branches. Also, in order to generate the eight parallel path metrics (i.e. first four branches) there are eight current metrics needed corresponding to "0" code bit and eight current metrics corresponding to "1".

Table 4.23 shows the transistor dimensions for the fixed-state symbol metric generator circuit of fig. 4.42. The bias voltage V_B is set to 1.5V to match the DC level of the SS output. The bias current I_B is 9μA which corresponds to a drain current (of either M_3 or M_4) range of about 2.5μA to 6μA for an input signal of ±100mV. The current mirrors amplify their input by two, which means that the current output range for $I\mu_{pi}$ is 5μA to 12μA. Assuming a current metric step of 1μA this range translates to the required 3-equivalent bit resolution. The circuit can be fine tuned to incorporate a slightly larger range, which corresponds to a metric step of 1.25μA if necessary, by adjusting I_B accordingly. The DC response of the circuit of fig. 4.42 is shown in fig. 4.43.

Fig. 4.44 shows a programmable transconductor used to generate symbol metrics for the variable branches in the first two modified tree levels (see section 3.4.2). The same circuit as that of the fixed-state metric generator of fig. 4.42 is used with the



Switch	W (μm)	L (μm)
S_1-S_4	4	0.6

Fig. 4.44 Variable-state programmable version of the circuit of fig. 4.42.

extra combination of switches S_1-S_4. The switches are controlled by voltages s_{pi} in order to realise the Euclidian distance as discussed in section 3.2.1 (see equation 3.8).

Four programmable symbol metric generator circuits are used resulting in a total of eight transconductors in the BMC. Each transconductor dissipates an average power of about 50μW.

CHAPTER 5

Conclusions and Future Work

Nowadays digital communication systems are widely used in our life in many applications such as mobile telephony, digital audio and video broadcasting and satellite communications. Especially in recent years the growth of wireless portable applications has reinforced the requirement for very low-power and small size devices. It has been demonstrated that using analogue circuit techniques in various digital signal processing and communication applications leads to optimum design solutions that meet the above requirements. This is partly due to the nature of signals in a practical communication system that are usually in the analogue domain (i.e. transmission waveforms); but mainly it is due to the flexibility of manipulating analogue signals using simple circuits.

In this thesis the first implementation of a new algorithm suitable for decoding convolutional codes has been presented. Analogue current-mode design techniques for the computational core of the decoder have been adopted due to their low power and simplicity. The importance of analogue VLSI decoding has been demonstrated and generally can be extended to other signal processing applications, where very large and power hungry digital sub-systems can be replaced by simple analogue counterparts.

In this chapter, a brief summary of the previous sections and results of the thesis will be outlined followed by future work.

5.1 Summary and Results

In chapter 1, a brief introduction and definition of the thesis objectives are outlined. A summary of the sections included and a list of conference publications resulting from this thesis are also presented.

Chapter 2 provides an introductory background on FEC concepts necessary to describe and present the three basic coding schemes with more emphasis on convolutional codes and their applications. A concise and fundamental introduction to digital communication systems was covered at the beginning of the chapter in order to illustrate the importance of channel coding. Convolutional codes have powerful correcting capabilities compared to other codes of similar complexity achieving large values of CG. They have been widely used in applications where high reliability and performance are necessary. The more recent Turbo codes sacrifice complexity and speed for higher coding performance approaching the theoretical limit. However, their implementation involves a number of iterative procedures that may significantly limit system performance in cases where moderate correcting capability is sufficient.

Chapter 3 covers a detailed description of the most commonly used convolutional decoding algorithm, namely VA, followed by the definition of the recently developed MFDA. Implementation considerations for the two algorithms in a comparative manner are also given. The MFDA is a hybrid FDA/VA incorporating merits of both algorithms while eliminating imposed limitations of each. These limitations are the exponentially increasing complexity of a FD with L and the required digital path memory (SSM) block in a VD which can be large and power hungry. Matlab® simulations using Simulink® modelling of various VD and MFD verify the potential advantages of an analogue MFD compared to two different realisations of the mixed-signal VD. The former compares favourably to a FSVD in terms of CG and to a BSVD in terms of implementation complexity, especially as K grows. A FSVD requires a large value of $L > 5K$ in order to achieve acceptable values of CG whereas a MFD can truncate its decoding window to $L < 4K$ for the same coding performance. On the other hand, a BSVD requires an $L = 4K$ but also requires an extra block (OD) which can be very complex for $K > 5$. Furthermore, extensive system-level modelling and simulations indicate that the performance of a MFD is almost insensitive to errors arising from the use of analogue circuit blocks except ACS feedback-loop gain errors. Although an accurate prediction of the latter errors is impossible, statistical averaging and worst case scenarios show that the performance of a MFD can significantly degrade especially as L grows. However, with the use of appropriate renormalisation techniques the loss in CG can be made dependent solely upon K and hence minimised. The above simulations were performed assuming a generic analogue circuit design approach using either current or voltage-mode circuitry providing essentially a very useful design guide to any particular approach.

Chapter 4 consists of two introductory sections covering the general advantages of the analogue current-mode approach and a literature overview of existing analogue decoders. The main section of this chapter presents the detailed design and first ever implementation of a $K = 3$, $L = 10$ hard-decision MFD. The decoder is intended as a proof of the MFDA principle and the critical computational core is realised in analogue current-mode as it would be in the soft-decision case. However, the remaining blocks are designed in digital, using standard blocks and components provided by the AMS portfolio. Since the input to a hard-decision decoder is binary, the mixed-signal implementation simplifies the design procedure. Nevertheless, the digital blocks used are certainly not optimised for the MFD and therefore further reduction in size and power dissipation can potentially be achieved. The MFD MIN reduces the required dynamic range and hence power consumption compared to a MAX implementation. The complete decoder, realised in a 3-metal 2-poly AMS CMOS 0.6μm technology, occupies a die area of 1.45mm^2 and dissipates an average power of 2.45mW from a single 3V supply when operating at 1Mb/s. Unlike most recent analogue decoders in the literature (i.e. Turbo style), the fabricated MFD has 100% yield and features a negligible performance loss. The RCC based circuits (i.e. ACS and WTA) described in this chapter are also very useful in a variety of signal processing and communications applications where power and complexity are of prime concern. The analogue core of the MFD is a pure current-mode design and including the SI memory cells it features a resolution of up to five equivalent-bits with a minimum metric step of 1μA. These figures accomplished using a mature CMOS technology certainly advance the state of the art especially in applications where the choice of the technology is limited by other factors (i.e. biomedical applications that require high voltage mature technologies). Also, it should be noted that the advantages of the analogue approach and hence the MFDA can be fully exploited in a larger, soft-decision MFD, where the performance limitations of existing digital decoders are enormous.

5.2 Future Work

Since the fabricated MFD proved fruitful, the following step would be the implementation of a fully analogue soft-decision decoder. In this case, the same analogue core can be used with slightly extended resolution in order to accommodate the higher dynamic range required. Simulation results in chapter 3 indicate that a soft-decision $K = 3$ MFD entails an increase of less than one equivalent-bit in the dynamic range of the ACS and WTA used for the hard-decision described in chapter 4. The only major differences in the design of a soft-decision decoder lie within the structure of the BMC and SS as explained in section 4.4. Using the circuits described in the latter section an almost entirely analogue convolutional decoder can be easily realised.

A larger constraint length MFD (i.e. $K = 5$) can also be designed using the same current-mode computational circuits. In order to minimise the ACS loop-gain errors and limit the path metric growth, suitable normalisation circuitry is required, which in turn reduces the dynamic range and resolution requirements to a minimum. As mentioned in chapter 3, for the construction of a soft-decision, $K = 5$ and $L = 20$ MFD incorporating an extra WTA block as normalisation circuitry, a range of about 5-equivalent bits is sufficient for no loss in coding performance. This translates to a current-mode range of 5-37μA that can be easily accommodated by the ACS and WTA circuits of the hard-decision MFD described in this thesis.

Also, where necessary, the speed of a MFD can be improved by using the N-step trellis technique. This method allows for multiple trellis levels to be processed in parallel at the expense of extra BMC complexity. For example, a $K = 5$, $L = 20$ decoder requires $L - K = 15$ sequential computations (i.e. clock cycles), using the conventional trellis implementation or $(L - K + 1) / 2 = 8$ cycles, using the 2-step trellis implementation, increasing the speed by almost a factor of two.

Finally, the MFD can be reconfigured to accommodate different modulation schemes, in order to be used in various wireless biomedical applications [14]-[17] requiring communications with external devices. In such applications, channel coding is in the early stages, and hence, increasing transmission reliability can have an enormous effect.

References

[1] C. E. Shannon, "A mathematical theory of communication", *Bell Syst. Tech. Jour.*, Vol. 27, pp. 379-423 (part I) and pp. 623-656 (part II), July 1948.

[2] A. J. Viterbi, "Convolutional codes and their performance in communication systems", *IEEE Trans. Commun. Techn.*, Vol.19, pp. 751-772, Oct. 1971.

[3] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd edition, Prentice-Hall, Upper Saddle River, NJ, 2001.

[4] G. C. Clark Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, NY, 1981.

[5] A. J. Viterbi and J. K. Omura, *Principles of Digital Communications and Coding*, McGraw-Hill, NY, 1979.

[6] S. Sridharan and L. R. Carley, " A 110 MHz 350 mW 0.6 μm CMOS 16-State Generalised-Target Viterbi Detector for Disk Drive Read Channels," *IEEE Jour. Solid State Circuits*, Vol. 35, No. 3, pp. 362-370, Mar. 2000.

[7] L. Jia *et al.*, "Design of a super-pipelined Viterbi decoder," *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Vol. 1, pp. 133-136, Jun. 1999.

[8] T. W. Matthews and R. P. Spencer, "An Integrated Analog CMOS Viterbi Detector for Digital Magnetic Recording," *IEEE J. Solid-State Circuits*, Vol. 28, No. 12, pp. 1294-1302, Dec. 1993.

[9] M. H. Shakiba, D. A. Johns and K. W. Martin, "An Integrated 200-MHz 3.3-V BiCMOS Class-IV Partial-Response Analog Viterbi Decoder," *IEEE J. Solid-State Circuits*, Vol. 33, No. 1, pp. 61-75, Jan. 1998.

[10] Kai He and G. Cauwenberghs, "Integrated 64-State Parallel Analog Viterbi Decoder", *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Vol. 4, pp. 761-764, May 2000.

[11] A. Demosthenous and J. Taylor, "A 100-Mb/s 2.8-V CMOS Current-Mode Analogue Viterbi Decoder", *IEEE J. Solid-State Circuits*, Vol.37, No.7, pp. 904-910, Jul. 2002.

[12] B. Zand and D. A. Johns, "High-Speed CMOS Analog Viterbi Detector for 4-PAM Partial-Response Signaling", *IEEE J. Solid-State Circuits*, Vol.37, No.7, pp. 895-903, Jul. 2002.

[13] K. Arabi and M. A. Sawan, "Electronic Design of a Multichannel Programmable Implant for Neuromuscular Electrical Stimulation," *IEEE Trans. Rehabilitation Eng.*, Vol. 7, pp. 204-214, Jun. 1999.

[14] Y. Neuvo, "The future is in wireless," *European Solid-State Circuits conf. (ESSIRC'2000),* Stockholm, Sweden, pp. 2-3, Sep. 2000.

[15] K. D. Wise *et al.*, "Wireless Implantable Microsystems: High-Density Electronic Interfaces to the Nervous System," *IEEE Proc.*, Vol. 92, No. 1, pp. 76-97, Jan. 2004.

[16] M. Ghovanloo and K. Najafi, "A Wideband Frequency-Shift Keying Wireless Link for Inductively Powered Biomedical Implants," *IEEE Trans. Circuits Syst. I*, Vol. 51, No. 12, pp. 2374-2383, Dec. 2004.

[17] M. Sawan, Yamu Hu and J. Coulombe, "Wireless Smart Implants Dedicated to Multichannel Monitoring and Microstimulation," *IEEE Circuits Syst. Magazine*, pp. 21-39, Mar. 2005.

[18] A. Demosthenous, C. Verdier and J. Taylor, "A New Architecture for Low Power Analogue Convolutional Decoders", *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 37-40, Jun. 1997.

[19] J. A. Heller, "Feedback Decoding of Convolutional Codes", *Advances in Commun. Syst.*, Vol.4, A. J. Viterbi (ed), NY: Academic, pp. 261-278.

[20] G. D. Forney, "The Viterbi Algorithm", *Proc. IEEE,* Vol.61, pp. 268-278, Mar. 1973.

[21] S. Hong and W. E. Stark, "Performance effects of using analogue memory in baseband signal processing system design," *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Vol. 1, pp. 703-706, May 2001.

[22] S. Hong and W. E. Stark, "Decoding Performance and Complexity Analysis for Analog and Digital Channel Decoders," *IEEE conf. Vehicular techn. (VTC)*, Vol. 2, pp. 1277-1281, May 2001.

[23] C. Toumazou, F. Lidgey and D. G. Haigh, *Switched-Currents: An Analogue Technique for Digital Technology*, Peter Peregrinus, London, 1993.

[24] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[25] K. Brayer and S. Natarajan "An Investigation of ARQ and Hybrid FEC-ARQ on an Experimental High Latitude Meteor Burst Channel", *IEEE Trans. Commun.*, Vol. 37, No. 11, Nov. 1989.

[26] R. J. Benice and A. H. Frey, Jr., "An Analysis of Retransmission Systems", *IEEE Trans. Commun.. Tech.*, pp. 135-145, Dec.1964.

[27] C. E. Shannon, "Communication in the Presence of Noise", *Proc. IRE*, Vol. 37, No. 1, pp. 10-21, Jan. 1949.

[28] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, Wiley, West Sussex, England, 2002.

[29] J. G. Proakis, *Digital Communications*, 2nd edition, McGraw-Hill, NY, 2001.

[30] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals", *IEEE Trans. Info. Theory*, Vol. IT-28, No 1, pp. 55-67, Jan 1982.

[31] H. Imai and S. Hirakawa, "A New Multilevel Coding Method Using Error-Correcting Codes", *IEEE Trans. Info. Theory*, Vol. IT-23, No 3, pp. 371-377, May 1977.

[32] V. K. Bhargava, "Forward Error Correction Schemes for Digital Communication Systems", *IEEE Commun. Mag.*, Vol. 21, pp. 11-19, Jan. 1983.

[33] B. Sklar, "Defining, Designing and Evaluating Digital Communication Systems", *IEEE Commun. Mag.*, Vol. 33, pp. 92-101, Nov. 1993.

[34] G. D. Forney Jr., "Generalized Minimum Distance Decoding", *IEEE Trans. Info. Theory*, Vol. IT-12, No 2, pp. 125-131, Apr. 1966.

[35] E. J. Weldon Jr., "Decoding Binary Block Codes on *Q*-ary Output Channels", *IEEE Trans. Info. Theory*, Vol. IT-17, No 6, pp. 713-718, Nov. 1971.

[36] D. Chase, "A Class of Algorithms for Decoding Block Codes With Channel Measurement Information", *IEEE Trans. Info. Theory*, Vol. IT-18, No 1, pp. 170-182, Jan. 1972.

[37] J. K. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis", *IEEE Trans. Info. Theory*, Vol. IT-24, No 1, pp. 76-80, Jan. 1978.

[38] Kiyohiro Furutani *et al.*, "A Built-In Hamming Code ECC circuit for DRAM's", *IEEE Jour. Solid State Circuits*, Vol. 24, No. 1, pp. 50-56, Feb. 1989.

[39] S. B. Wicker and J. Kosmach, "Advances in Soft Decision Reed-Solomon Decoding for Wireless Applications", *IEEE Conf. Commun. Computers & Signal Processing*, Vol. 2, pp. 878-882, Aug. 1997.

[40] R. D. Cideciyan, E. Eleftheriou and S. Tomasin, "Performance Analysis of Magnetic Recording Systems", *IEEE Int. Conf. Commun (ICC)*, Vol. 9, Jun. 2001.

[41] R. Wood, "Magnetic and Optical Storage Systems Opportunities for Communication Technology", *IEEE Int. Conf. Commun (ICC)*, Vol. 3, pp. 1605-1612, Jun. 1989.

[42] J. C. Huang, C. M. Wu, M. D. Shieh and C. H. Wu, "An Area-Efficient Versatile Reed-Solomon Decoder for ADSL", *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Vol. 1, pp. 517-520, Jun. 1999.

[43] J. P. Odenwalder, *Error Control Coding Handbook*, Linkabit Corporation, SD, California, July 1976.

[44] J. A. Heller, "Short Constraint Length Convolutional Codes", Jet Propulsion Lab., California Inst. Technol., *Space Programs Summary 37-54*, Vol. 3, pp. 171-177, Oct./Nov. 1968.

[45] D. J. Costello, Jr., "Free Distance Bounds for Convolutional Codes", *IEEE Trans. Info. Theory*, Vol. IT-20, No 3, pp. 356-365, May 1974.

[46] J. L. Massey and M. K. Sain, "Inverse of Linear Sequential Circuits", *IEEE Trans. Comput.*, Vol. C-17, pp. 330-337, Apr. 1968.

[47] D. G. Daut et al., "New Short Constraint Length Convolutional Code Construction for Selected Rational Rates", *IEEE Trans. Info. Theory*, Vol. IT-28, pp. 793-799, Sep. 1982.

[48] E. Paaske, "Short Binary Convolutional Codes with Maximal Free Distance for rates 2/3 and 3/4", *IEEE Trans. Info. Theory*, Vol. IT-20, pp. 683-689, Sep. 1974.

[49] K. J. Larsen, "Short Convolutional Codes with Maximal Free Distance for rates 1/2, 1/3 and 1/4", *IEEE Trans. Info. Theory*, Vol. IT-19, pp. 371-372, May 1973.

[50] J. B. Cain, G. C. Clark and J. M. Geist, "Punctured Convolutional Codes of Rate $(n-1)/n$ and Simplified Maximum Likelihood Decoding", *IEEE Trans. Info. Theory*, Vol. IT-25, No. 1, pp. 97-100, Jan 1979.

[51] G. D. Forney, *Concatenated Codes*, MIT Press, Cambridge, MA, 1966.

[52] J. L. Ramsey, "Realization of Optimum Interleavers", *IEEE Trans. Info. Theory*, Vol. IT-16, No. 3, pp. 338-345, May 1970.

[53] G. D. Forney, "Burst-Correcting Codes for the Classic Bursty Channel", *IEEE Trans. Commun. Tech.*, Vol. COM-19, pp. 772-781, Oct. 1971.

[54] L. H. C. Lee, *Convolutional Coding: Fundamentals and Applications*, Artech House, Norwood, MA, 1997.

[55] http://nssdc.gsfc.nasa.gov

[56] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*, IEEE press, 1994.

[57] J. H. Yuen *et al.*, "Modulation and Coding for Satellite and Space Communications", *IEEE Proc.*, Vol. 78, No. 7, pp. 1250-1266, July 1990.

[58] Takashi Lida, *Satellite Communications: System and its Design Technology*, Ohmsha, Ios press, Oxford, England, 2000.

[59] W. W. Wu *et al.*, "Coding for Satellite Communications", *IEEE Jour. Selected Areas Commun.*, Vol. SAC-5, No. 4, pp. 724-748, May 1987.

[60] J. Uddenfeldt, "Digital Cellular–Its Roots and Its Future", *IEEE Proc.*, Vol. 86, No. 7, pp. 1319-1324, July 1998.

[61] J. Eberspacher, H. J. Vogel and C. Bettstetter, *GSM Switching, Services and Protocols*, 2nd edition, Wiley & Sons, West Sussex, England, 2001.

[62] J. Hagenauer, "Rate Compatible Punctured Convolutional RCPC Codes and their Applications", *IEEE Trans. Commun.*, Vol. COM-36, pp. 389-400, Apr. 1988.

[63] U. Reimers, *Digital Video Broadcasting: The International Standard for Digital Television*, Springer-Verlag, Berlin, Germany, 2001.

[64] W. Hoeg and T. Lauterbach, *Digital Audio Broadcasting: Principles and Applications*, Wiley & Sons, West Sussex, England, 2001.

[65] D. J. Costello, Jr., *et al.*, "Applications of Error-Control Coding", *IEE Trans. Info. Theory*, Vol. 44, No. 6, pp. 2531-2560, Oct. 1998.

[66] F. Q. Wang and D. J. Costello, Jr., "New Rotationally Invariant Four-Dimensional Trellis Codes", *IEEE Trans. Info. Theory*, Vol. IT-42, No. 1, pp. 291-300, Jan. 1996.

[67] G. D. Forney, Jr., *et al.*, "The V.34 High-Speed Modem Standard", *IEEE Commun. Magazine*, Vol. 43, pp. 28-33, Dec. 1996.

[68] C. Berou, *et al.*, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", *IEEE Proc. Int. Conference Commun. ICC'93*, pp. 1064-1070, May 1993.

[69] M. C. Valenti and J. Sun, *Turbo Codes*, Ch. 12 of *Handbook of RF and Wireless Technologies*, (editor F. Dowla), Newnes, Oxford 2004.

[70] L. Perez, J. Seghers and D. J. Costello, "A Distance Spectrum Interpretation of Turbo Codes", *IEEE Trans. Info. Theory*, Vol. 42, No. 6, pp. 1698-1708, Nov. 1996.

[71] D. J. Costello, Jr. *et al.*, "Turbo Decoding with Tail-Biting Trellises", *IEEE URSI Int. Symp. Signals Syst. Electron.*, (ISSSE'98), pp. 343-348, Oct. 98.

[72] R. Koetter and A. Vardy, "The Structure of Tail-Biting Trellises: Minimality and Basic Principles", *IEEE Trans. Info. Theory*, Vol. 49, No. 9, pp. 2081-2105, Sep. 2003.

[73] S. Pietrobon, "A Turbo/MAP Decoder for use in Satellite circuits", *Int. conf. Info. Commun. Signal Proc.*, (ICICS'97), Vol. 1, pp. 427-431, Sep. 1997.

[74] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications", *IEEE Global Telecom. Conf.*, (GLOBECOM'89), Vol. 3, pp. 1680-1686, Nov. 1989.

[75] Engling Yeo *et al.*, "A 500-Mb/s Soft-Output Viterbi Decoder", *IEEE J. Solid-State Circuits*, Vol.38, No.7, pp. 1234-1241, Jul. 2003.

[76] L. N. Lee, *et al.*, "Application and Standardization of Turbo codes in Third-Generation High-Speed Wireless Data Services", *IEEE Trans. Vehicular Tech.*, Vol. 49, No. 6, pp. 2198-2207, Nov. 2000.

[77] J. B. Berner and K. S. Andrews, "Deep Space Network Turbo Decoder Implementation", *IEEE Aerospace Conf. Proc.*, Vol. 3, pp. 1149-1157, Mar. 2001.

[78] M. C. Valenti, "Inserting Turbo Code Technology into the DVB Satellite System", *IEEE Military Commun. Conf. Proc.*, (MCC'00), Vol. 2, pp. 650-654, Oct. 2000.

[79] J. M. Wozencraft and B. Reiffen, *Sequential decoding*, MIT Press, Cambridge, MA, 1961.

[80] R. M. Fano, "A Heuristic Discussion of Probabilistic Decoding", *IEEE Trans. Info. Theory*, Vol. IT-9, pp. 64-74, Apr. 1963.

[81] F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack", *IBM J. Res. Develop.*, Vol. 13, pp.675-685, Nov. 1969.

[82] J. B. Cain and G. C. Clark, "Some Results on the Error Propagation of Convolutional Feedback Decoders", *IEEE Trans. Info. Theory*, Vol. IT-18, pp. 681-683, Sep. 1972.

[83] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE. Trans. Info. Theory*, Vol. IT-13, pp. 260-269, Apr. 1967.

[84] H. Loo, "Implementing the Viterbi Algorithm" *IEEE Signal Processing Mag.*, Vol.12, pp. 42-52, Sep. 1995.

[85] M. H. Shakiba, D. A. Johns and K. W. Martin, "BiCMOS Circuits for Analog Viterbi Decoders", *IEEE Trans. Circuits Syst. II*, Vol.45, pp. 1527-1537, Dec. 1998.

[86] A. Demosthenous and J. Taylor, "Low-Power CMOS and BiCMOS Circuits for Analogue Convolutional Decoders", *IEEE Trans. Circuits Syst. II*, Vol.46, pp. 1077-1081, Aug. 1999.

[87] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE press, NJ, 1999.

[88] J. A. Heller, "Viterbi Decoding for Satellite and Space Communications", *IEEE Trans. Commun. Techn.*, Vol. COM-19, No. 5, pp. 835-848, Oct. 1971.

[89] G. Fettweis and H. Meyr, "Parallel Viterbi Algorithm Implementation: Breaking the ACS-Bottleneck", *IEEE Trans. Commun.*, Vol. 37, No. 8, pp. 785-790, Aug. 1989.

[90] I. Kang and A. N. Wilson "Low-Power Viterbi Decoder for CDMA mobile terminals", *IEEE J. Solid-State Circuits*, Vol.33, No.3, pp. 473-481, Mar. 1998.

[91] Y. Zhu and M. Benaissa, "A Novel ACS Scheme for Area-Efficient Viterbi Decoders", *Int. Symp. Circuits Syst. (ISCAS)*, Vol. 2, pp. 264-267, May 2003.

[92] Y. N. Chang, H. Suzuki and K. K. Parhi, "A 2-Mb/s 256-State 10mW Rate-1/3 Viterbi Decoder", *IEEE J. Solid-State Circuits*, Vol.35, No.6, pp. 826-834, Jun. 2000.

[93] A. K. Yeung and J. M. Rabaey, "A 210Mb/s Radix-4 Bit-level Pipelined Viterbi Decoder", *IEEE Int. Conf. Solid-State Circuits (ISSCC)*, pp. 88-90, Feb. 1995.

[94] G. Ungerboeck *et al.*, "VLSI Architectures for Metric Normalization in the Viterbi Algorithm", *IEEE Int. Conf. Commun. (SUPERCOM/ICC)*, pp. 1723-1728, Apr. 1990.

[95] L. I. Qiao et al., "VLSI Implementation of a High-Speed and Low-power Punctured Viterbi Decoder", *IEEE conf. Computers, Commun., Control and Power Engineering (TENCON)*, Vol. 2, pp. 1205-1208, Oct. 2002.

[96] C. M. Rader, "Memory Management in a Viterbi Algorithm", *IEEE Trans. Commun.*, Vol. 29, No. 8, pp. 1399-1401, Sep. 1981.

[97] G. Feygin and P. G. Gulak, "Architectural Tradeoffs for Surviving Sequence Memory Management in Viterbi Decoders", *IEEE Trans. Commun.*, Vol. 41, No. 3, pp. 425-429, Mar. 1993.

[98] P. J. Black and T. H.-Y. Meng, "Hybrid Survivor Path Architectures for Viterbi Decoders", *IEEE Int. Conf Acoustics, Speech Signal Proc.*, Vol. 1, pp. 433-436, Apr. 1993.

[99] R. J. McEliece and I. M. Onyszchuk, "Truncation Effects in Viterbi Decoding", *IEEE Military Commun. Conf. (MILCOM)*, Vol. 2, pp. 541-545, Oct. 1989.

[100] J. Lazzaro *et al.*, "Winner-take-all networks of O(N) complexity", *Advances in Neural Info. Processing Syst.*, I. D. S Touretzky, Ed., San Mateo, CA: Morgan Kaufmann, 1989, pp. 703-711.

[101] A. Demosthenous, S. Smedley and J. Taylor, "A CMOS Analogue Winner Take-All Network for Large Scale-Applications", *IEEE Trans. Circuits Syst. I*, Vol. 45, pp. 360-364, Mar. 1998.

[102] S. Acampora and R. P. Gilmore, "Analog Viterbi decoding for high speed digital satellite channels," *IEEE Trans. Commun.*, Vol. COM-26, pp. 1463-1470, Oct.1978.

[103] A. Demosthenous and John Taylor, "Effects of signal-dependant errors on the performance of switched-current Viterbi decoders," *IEEE Trans. Circuits Syst. I*, vol. 48, pp. 1225-1228, Oct. 2001.

[104] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, "Matching Properties of MOS transistors," *IEEE J. Solid-State Circuits*, Vol. 24, No. 1, pp. 1433-1439, Oct. 1989.

[105] A. Demosthenous and John Taylor, "High-Speed Replicating Current Comparators for Analog Convolutional Decoders," *IEEE Trans. Circuits Syst. I*, vol. 47, No. 12, pp. 1405-1412, Dec. 2000.

[106] J. Hagenauer, M. Moerz and A. Schaefer, "Analog Decoders and Receivers for High Speed Applications," *IEEE Int. seminar Broadband Commun. (ATN)*, pp. 3.1-3.8, Feb. 2002.

[107] F. Lustenberger, *On the Design of Analog VLSI Iterative Decoders*, PhD dissertation, Swiss Federal Institute of Technology (ETH No. 13879), Zurich, Nov. 2000.

[108] R. P. Spencer, "Simulated Performance of Analog Viterbi Detectors," *IEEE j. selected areas in commun.*, Vol. 10, No. 1, pp.277-287, Jan 1992.

[109] V. C. Gaudet and P. G. Gulac, "A 13.3-Mb/s 0.35-μm CMOS Analog Turbo Decoder IC With a Configurable Interleaver," *IEEE J. Solid-State Circuits*, Vol. 38, No. 11, pp. 2010-2015, Nov. 2003.

[110] C. Winstead *et al.*, "CMOS Analog MAP Decoder for (8,4) Hamming Code," *IEEE J. Solid-State Circuits*, Vol. 39, No. 1, pp. 122-131, Jan. 2004.

[111] C. Winstead and C Schlegel "Analog Decoding: State of the Art," *IEEE Int. Symp. Spread Spectrum Systems and Techniques*, pp. 503-510, Aug. 2004.

[112] D. Vogrig *et al.*, "A 0.35-μm CMOS Analog Turbo Decoder for the 40-bit Rate 1/3 UMTS Channel Code" *IEEE J. Solid-State Circuits*, Vol. 40, No. 3, pp. 753-762, Mar. 2005.

[113] C. Winstead *et al.*, "Low-Voltage CMOS Circuits for Analogue Iterative Decoders," *IEEE Trans. Circuits Syst. I*, vol. 53, No. 4, pp. 829-841, Apr. 2006.

[114] R. W. Wood, "Magnetic Megabits," *IEEE Spectrum*, pp. 32-38, May 1990.

[115] R. Philpott *et al.*, "A 7Mb/sec (65 MHz), Mixed-Signal, Magnetic Recording Channel DSP Using Partial Response Signalling with Maximum Likelihood Detection," *IEEE Conf. dig. Custom Integrated Circuits*, pp. 10.4.1-10.4.4, Aug. 1993.

[116] R. W. Wood and D. A. Patersen, "Viterbi Detection of Class-IV Partial Response on a Magnetic Recording Channel," *IEEE Trans. Commun.*, Vol. COM-34, pp.454-461, May 1986.

[117] H. A. Loeliger *et al.*, "Probability Propagation and Decoding in Analog VLSI," *Trans. Info. Theory*, Vol. 47, No 2, pp. 837-843, Feb. 2001.

[118] H. A. Loeliger *et al.*, "Decoding in Analog VLSI," IEEE Commun. Mag., pp. 99-101, Apr 1999.

[119] F. Lustenberger *et al.*, "All-Analog Decoder for a Binary (18,9,5) Tail-Biting Trellis Code," *IEEE ESSIRC*, pp. 362-365, Duisburg, Germany, Sep. 1999.

[120] M. Moerz *et al.*, "An analog 0.25-μm BiCMOS tailbiting MAP decoder," *IEEE Int. Solid-State Circuits Conf.*, pp. 356-357, Feb. 2000.

[121] P. E. Allen and D. R. Holberg, "*CMOS Analog Circuit Design,*" 2nd edition, Oxford University press, NY, 2002.

[122] P. M. Figueiredo and J. C. Vital, "Low Kickback Noise Techniques for CMOS Latched Comparators," *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Vol. 1, pp. 537-540, May 2004.

[123] R. G. Carvajal *et al.*, "High-Speed High-Precission Voltage-Mode MIN/MAX Circuits in CMOS Technology," *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Vol. 5, pp. 13-16, May 2000.

[124] T. Yamakawa, "A Fuzzy Inference Engine in Nonlinear Analog Mode and its Applications to Fuzzy Logic Control," *IEEE Trans. Neural Networks*, vol. 4, pp. 496-522, May 1993.

[125] M. H. Shakiba, D. A. Johns and K. W. Martin, "BiCMOS Circuits for Analog Viterbi Decoders," *IEEE Trans. Circuits Syst. II*, vol. 45, No. 12, pp. 1527-1537, Dec. 1998.

[126] H. Chaoui, "CMOS Multiple Input Winner Takes All Circuit," *IEEE Electron. Lett.* vol. 31, No. 22, pp. 1915-1916, Oct. 1995.

[127] I. E. Opris, "Rail-to-Rail Multiple-Input Min/Max Circuit," *IEEE Trans. Circuits Syst. II*, vol. 45, No. 1, pp. 137-140, Jan. 1998.

[128] B. Maundy, "Min/Max Circuit for Analog Convolutional Decoders," *IEEE Trans. Circuits Syst. II*, vol. 48, No. 8, pp. 802-806, Aug. 2001.

[129] C. Y. Huang and B. D. Liu, "Current-Mode Multiple Input Maximum Circuit for Fuzzy Logic Controllers," *IEEE Electron. Lett.* vol. 30, No. 23, pp. 1924-1925, Nov. 1994.

[130] I. Baturone *et al.*, "Current-Mode Multiple Input Max Circuit," *IEEE Electron. Lett.* vol. 30, No. 9, pp. 678-680, Apr. 1994.

[131] S. Keawconthai *et al.*, "An Analogue Current-Mode Multiple Input Max/Min Circuit," *IEEE ICSP Proc.*, pp. 507-510, 2004.

[132] M. Sasaki *et al.*, "Fuzzy Multiple-Input Maximum and Minimum Circuits in Current Mode and Their Analyses Using Bounded-Difference Equations", *IEEE Tans. Computers*, vol. 39, No. 6, pp 768-774, June 1990.

[133] T. Serrano-Gotarredona and B. Linares-Barranco, "A High-Precission Current-Mode WTA-MAX Circuit With Multichip Capability," *IEEE J. Solid-State Circuits*, Vol. 33, No. 2, pp. 280-286, Feb. 1998.

[134] A. Fish, V. Milrud and O. Yadid-Pecht, "High-Speed and High-Precission Current Winner-Take-All Circuit," *IEEE Trans. Circuits Syst. II*, vol. 52, No. 3, pp. 131-135, Mar. 2005.

[135] J. Ramírez-Angulo *et al.*, "Low-Voltage High-Performance Voltage-Mode and Current-Mode WTA Circuits Based on Flipped Voltage Followers," *IEEE Trans. Circuits Syst. II*, vol. 52, No. 7, pp. 420-423, July 2005.

[136] Ramon Gonzalez *et al.*, "The Flipped Voltage Follower: A Useful Cell for Low-Voltage Low-Power Circuit Design," *IEEE Trans. Circuits Syst. I*, vol. 52, No. 7, pp. 1276-1291, July 2005.

[137] A. Hastings, *The Art of Analog Layout*, Prentice-Hall, NJ, 2001.

[138] Yeong-Sheng Lee and K. W. Martin, "A Switched Capacitor Realisation of Multiple FIR Filters on a Single Chip," *IEEE J. Solid-State Circuits*, Vol. 23, No. 2, pp. 536-542, Apr. 1988.

[139] Tsung-Sum Lee and Chi-Chang Lu, "A 1.5-V 50-MHz Pseudodifferential CMOS Sample-and-Hold Circuit With Low Hold Pedestal," *IEEE Trans. Circuits Syst. I*, vol. 52, No. 9, pp. 1752-1757, Sep. 2005.