Novel Architectures for Miniaturised Low-Power Convolutional Decoders Using Current-Mode Analogue Circuit Techniques

by

Andreas Christodoulou Demosthenous

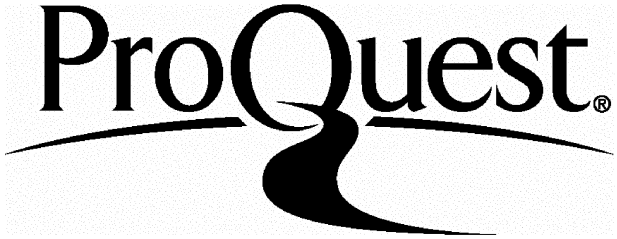
A thesis submitted for the degree of

Doctor of Philosophy

University College London

April 1998

Department of Electronic and Electrical Engineering
University College London
Torrington Place
London, WC1E 7JE



To my parents

Abstract

Convolutional decoders have long been important in applications where very noisy channels are encountered, such as satellite communications. The technique has recently become much more significant with the advent of new application areas such as mobile radio systems of various types and digital magnetic disk recording. Convolutional decoders usually employ the Viterbi algorithm (VA), implemented in software or digital hardware, depending on the requirements and constraints of the application. In an attempt to reduce the power dissipation and size of Viterbi decoders, without sacrificing speed, analogue circuits have recently been employed for the realisation of certain sections of these systems. Such hybrid analogue/digital Viterbi decoders are found in today's state-of-the-art computer disk drives. However, most of the analogue Viterbi realisations reported so far employ a simplified form of the VA, and hence are not suitable for decoding convolutional codes.

This thesis describes the application of the analogue current-mode approach to the design of convolutional decoders, employing both the VA in standard hybrid analogue/digital architectures, and in addition, a new convolutional decoding algorithm, which is called the *modified feedback decoding algorithm* (MFDA). The MFDA enables a convolutional decoder to be constructed almost entirely from analogue components, with only negligible loss of coding gain compared to a Viterbi decoder, thus offering the potential for construction of very small and economical systems. The use of analogue current-mode techniques in this application offers a more attractive trade-off in terms of speed, size and power dissipation than the conventional use of analogue voltage-mode techniques, as well as allowing operation at 2.8V or less. Furthermore, stemming from the work on decoding, this thesis also describes the design, implementation and evaluation of a CMOS winner-take-all network (WTA) which is well suited to applications requiring large WTA systems where operating speed and resolution are important parameters [e.g., vector quantisation (VQ)].

Acknowledgements

I would like to thank my supervisor Dr John Taylor for his contributions to my thesis and for employing me as Research Fellow for the next three years to continue research in this topical field of analogue convolutional decoders.

Many thanks to my wife Angela who has shared the ups and downs of my life as a graduate student these past few years. Words cannot express my appreciation for her being in my life.

I am grateful to my parents who have been a constant support and encouragement. I know they are very proud of me. Many thanks to my godfather Costas Pattihis who has shared the cost of funding my PhD with my parents. Thanks also to my brothers Demos and Pavlos for lending me their computers when I need them.

Thanks to Sean Smedly for his help on some of the circuit design work and for showing me how to use the Cadence layout tool, and to Mike Brent at Imperial College for always sorting out the various problems with the Cadence tool. Thanks also to Celiné Verdier for her contributions to the simulations of the MFDA. Finally, I am grateful to all the students in room 621 in the department of Electrical and Electronic Engineering at UCL, and especially Rahim for their support and friendship.

Table of Contents

List of Figures and Tables

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Glossary of Terms

2-WTA	2-input Winner-Take-All
A/D	Analogue-to-Digital
ACS	Add-Compare-Select
ACSU	Add-Compare-Select Unit
ANN	Artificial Neural Network
ARQ	Automatic Repeat reQuest
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BMC	Branch Metric Computer
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CFT	Clock FeedThrough
CG	Coding Gain
DMC	Discrete Memoryless Channel
DSP	Digital Signal Processor
DVB-S	Digital Video Broadcasting–via Satellite
EPR	Extended Partial-Response
FEC	Forward Error Correction
ISI	InterSymbol Interference
MFD	Modified Feedback Decoder
MFDA	Modified Feedback Decoding Algorithm
ML	Maximum Likelihood
MLSD	Maximum likelihood Sequence Detection

OD	Output Decision
RS	Reed-Solomon
PR	Partial-Response
PRS	Partial-Response Signalling
S/H	Sample-and-Hold
S/N	Signal-to-Noise
SS	Symbol Storage
SSM	Storage Survivor Memory
T/H	Track-and-Hold
TMC	Trellis-Coded Modulation
VA	Viterbi Algorithm
VQ	Vector Quantisation
WTA	Winner-Take-All

Chapter

1

Introduction

CONVOLUTIONAL encoding/decoding systems have long been very important components of digital communication systems; they are used to combat channel distortions. This is especially so in situations involving very noisy channels, where high levels of error-correction are required. Examples of such noisy environments occur in satellite communication systems, digital cellular telephony and digital modems. The popularity of convolutional codes stems from the existence of several decoding techniques that provide a range of complexity versus performance options and, most importantly, an optimum decoding algorithm that lends itself particularly well to *soft-decision* decoding and can be implemented with reasonable complexity. Thus, in noisy environments the convolutional approach provides a more economical solution than the competing block coding techniques for similar levels of error-correction.

The most commonly employed decoding method for convolutional codes is the *Viterbi algorithm* (VA) [1], [2], which is a special application of dynamic programming to communication theory. The VA provides a practical method for realising *maximum-likelihood sequence detection* (MLSD) and practical realisations generally employ software or digital hardware, depending on the requirements of the application. Single chip digital Viterbi decoders are commercially available operating at about 25Mbit/s output data rate and dissipating up to 0.8W of power [3]. In outline, the VA seeks to

determine the most-likely transmitted sequence having the minimum distance from the received signal, as compared to all other possibly transmitted sequences. The objective function to be minimised depends on the error criterion and is commonly the squared Euclidean distance.

In addition to convolutional codes, the VA is frequently used in other applications requiring optimum estimation of a digital sequence. For example, the VA is employed in the realisation of detectors for *partial-response signalling* (PRS) in digital magnetic recording systems [4]-[9]. Employing PRS in combination with MLSD in the new generation of hard disks as opposed to the traditional symbol-by-symbol detection, is the primary reason for the tremendous increase in storage capacity [10] offered by these devices.

Although practical realisations of Viterbi decoders generally employ software or digital hardware, there are considerable advantages in realising certain sections of these decoders using analogue circuits, i.e., to form a *hybrid analogue/digital* system. This is particularly true for applications demanding high-speed, low-power and small size such as digital magnetic recording systems and certain types of high-speed data transmission systems [11]. In such applications, analogue circuits have a number of potential advantages over their digital equivalents, mostly stemming from the fact that a high degree of accuracy is not required. For small to medium size decoders, an accuracy of about 6 equivalent bits is adequate [4], [12] and this is within the capability of modern analogue circuits and systems. In addition, using analogue techniques, a soft Viterbi decoder can be constructed without the need for an analogue-to-digital (A/D) converter, which would be required in a digital realisation. Also, analogue circuits generally allow more effective trade-offs to be obtained between speed, size and power dissipation than their digital counterparts. Recently, the magnetic recording industry has shown a lot of interest in pursuing this viable alternative and many state-of-the-art computer disk drives employ analogue Viterbi decoders[1] operating in the order of a few hundreds of Mbit/s [6], [7], [9].

Most of the integrated analogue Viterbi decoder realisations reported so far have been restricted to class-IV [5]-[8] PRS for digital magnetic recording systems. These decoders use a simplified form of the VA and are not applicable to general decoding applications,

[1] Hybrid analogue/digital Viterbi decoders are widely known as analogue Viterbi decoders in the literature.

such as decoding convolutional codes. To date the analogue sections of these decoders have been implemented using exclusively *voltage-mode* analogue circuit techniques in CMOS or BiCMOS technologies. A more recent publication has reported an analogue Viterbi decoder for general applications also employing voltage-mode analogue circuits and realised in BiCMOS technology [13]. Although, the potential operating speed of this approach is very high, the power dissipation is also very high, and 5V supply rails are required for proper operation.

1.1 Research Objectives

Scaling/reliability considerations are driving CMOS (and BiCMOS) technologies toward the sub-micron regime as compared with today's 0.35μm state-of-the-art devices operating at 3.3V or less. In the future evolution of CMOS (and BiCMOS) processes scaling of the voltage supply rails will become a crucial issue and voltage reduction down to as low as 1V or less is likely to be seen [14]. However, these factors make it considerably more difficult to design analogue circuits than digital circuits, and analogue voltage-mode circuits with high linearity and wide dynamic range are especially difficult. Therefore, over the last few years a class of analogue circuits in which current rather than voltage is the principal signal medium has been receiving considerable attention. *Current-mode* analogue circuits offer many potential advantages compared to their voltage-mode equivalents [15]. Wide bandwidth, low-voltage operation, wide dynamic range, and easy manipulation of signals are the outstanding features of this approach. In addition, in many applications cheap single-poly (digital) CMOS processes can be used for manufacturing since usually no MOS capacitors are required [16].

The application of the current-mode approach to the implementation of general classes of analogue Viterbi decoder is one of the objectives of this thesis. The use of this approach in this application not only promises high-speed operation, but also very low-power dissipation, small size, and scaling of the supply rails down to 2.8V or less. These features are particularly important for many modern and future types of communication systems where low-voltage, high-speed operation and mobility are simultaneous requirements (e.g., high-speed data transmission via satellite to remote users and portable computer disk drives).

Generally, in an analogue Viterbi decoder the digital section occupies most of the silicon area and dissipates most of the total power. An obvious solution to this problem is to

construct an *all-analogue* Viterbi decoder; in other words a Viterbi decoder realised completely in the analogue domain. However, replacing the digital section of a Viterbi decoder with analogue components and subsystems is not a practical solution because the operations performed there do not lend themselves to implementation in the analogue domain. Thus, another objective of this thesis is the development and performance evaluation of a new practical convolutional decoding technique that, unlike the conventional VA, enables a convolutional decoder to be constructed *almost entirely* from analogue components and subsystems. The use of this algorithm which in this thesis is referred to as the *modified feedback decoding algorithm* (MFDA) makes possible a much more flexible trade-off between speed, size and power dissipation than is possible with the hybrid analogue/digital VA approach. The MFDA approach enables a convolutional decoder to be constructed which is very much smaller and less power hungry than was hitherto possible. Such a convolutional decoder would be very desirable for many applications where small size and low-power dissipation are crucial parameters (e.g., certain types of future wireless systems [17]). The design of current-mode analogue circuits for the realisation of the MFDA is also investigated in this thesis.

The advantages of the current-mode approach can be extended to other digital communication subsystems such as *vector quantisation* (VQ) systems [18]. VQ is a modern data compression technique, which is currently receiving considerable attention because it enables high levels of compression to be achieved. Data compression is essential for reducing the data transmission requirements and storage costs in applications such as high-definition television (HDTV), multimedia systems and image database management. The operation of a vector quantiser is quite similar to that of a convolutional decoder in the sense of seeking to find the most-likely transmitted sequence having the minimum distance from the received sequence. Furthermore, there are many architectural similarities between these systems, especially when implementation is done in the analogue domain. An important building block of both systems is the *winner-take-all* (WTA) network [19]. However, vector quantisers in contrast to convolutional decoders, require not only complex WTA systems, but also WTA systems with a large number of inputs and a high level of accuracy. Therefore, the final objective of this thesis is the design, implementation and evaluation of a high-speed scalable analogue WTA network based on current-mode circuit principles. The modular architecture, compactness, high-speed operation, low-power consumption and high

accuracy of the presented system make it particularly well suited for use in large-scale analogue VQ realisations.

Based on the work of this thesis three years funding has been obtained from the UK Engineering and Physical Science Research Council (EPSRC) with effect from 1st April 1998 to continue research.

1.2 Published Papers

The research has resulted in a number of publications in international journals and conferences. These are listed in chronological order below:

1. C. Verdier, A. Demosthenous, J. Taylor, and M. Wilby, "An integrated analogue convolutional decoder based on the Hamming neural classifier," in *Proc. Neural Networks and their Applications (NEURAP)*, Marseilles, France, Mar. 1996, pp. 150-155.

2. A. Demosthenous, J. Taylor, and S. Smedley, "A high-speed scalable CMOS winner-take-all network," in *Proc. Int. Conf. Artificial Neural Networks (ICANN),* Bochum, Germany, Jun. 1996, pp. 370-375, Jun. 1996.

3. A. Demosthenous and J. Taylor, "Current-mode approaches to implementing hybrid analogue/digital Viterbi decoders," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Rhodes, Greece, Oct. 1996, pp. 33-36.

4. A. Demosthenous, R. Akbari-Dilmaghani, J. Taylor, and S. Smedley, "Enhanced modular CMOS winner-take-all network," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Rhodes, Greece, Oct. 1996, pp. 402-405.

5. A. Demosthenous, C. Verdier, and J. Taylor, "A new architecture for low-power analogue convolutional decoders," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Hong Kong, Jun. 1997, pp. 37-40.

6. 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.

7. A. Demosthenous and J. Taylor, "BiCMOS add-compare-select units for Viterbi decoders," due for publication in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Monterey, CA. May 1998.

8. A. Demosthenous and J. Taylor, "A winner-take-all network for large-scale analogue vector quantisers," due for publication in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Monterey, CA, May 1998.

9. A. Demosthenous and J. Taylor, "Low-power BiCMOS circuits for convolutional decoders" submitted to *IEEE Trans. Circuits Syst. I*, Apr. 1998.

Chapter

2

Error-Control Coding and Convolutional Codes

A primary concern in any digital communication system is the reliability with which transmitted information can be received at the destination. Whenever the received information is not the same as the transmitted information over some time interval, errors are said to have occurred. If the average error rate of the received information is above the desired level specified by the system, some means of improving communication reliability must be introduced. One solution is to improve the physical transmission medium. For example, if the communication link was from a satellite to a terrestrial aerial, the satellite radiated power or the terminal antenna gain could be increased. However, a more economical solution is to introduce *error-control coding* into the system [20], [21]. The usefulness of coding was first demonstrated by Shannon [22]. In 1948 he showed that by proper encoding of the information to be transmitted, errors introduced by a noisy channel could be essentially reduced to any desired level without sacrificing the information transmission rate.

This chapter provides an insight into error-control coding to overcome channel distortions with emphasis on convolutional coding. The chapter begins with a description of the important features of a modern digital communication system that incorporates error-control processing. It then describes the different modern error-control approaches that

are encountered in practice. Block and convolutional codes are then introduced, followed by error-control techniques for combating burst errors. Following this, the fundamental structures and basic properties of convolutional codes are presented. Finally, the chapter ends with a survey of the key practical applications of convolutional codes to modern communication systems.

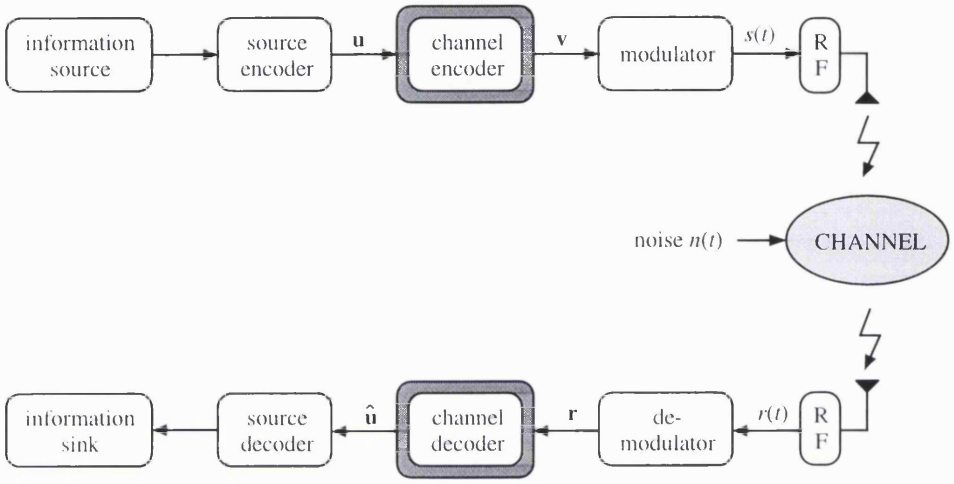


Figure 2.1 Block diagram of a typical digital communications system.

2.1 Elements of a Digital Communication System

The essential features of a typical digital communications system are shown in the block diagram of Fig. 2.1. The *information source* which is modelled probabilistically could be either discrete (e.g., an alphanumeric keyboard generating discrete symbols that correspond to a digital alphabet), or continuous (e.g., a speech signal produced by a microphone which is continuous in amplitude and time). The *source encoder* accepts the source outputs and produces a digital sequence called the *information sequence* **u**, which represents the source output in the best possible way (with respect to some distortion measure). Most frequently, the information sequence **u** consists of binary symbols called *bits* (i.e., a "0" or a "1") but sometimes it can be nonbinary with symbols from a higher order digital alphabet[1]. Actually, the source encoder performs data compression, i.e., reduction of the number of symbols per unit time required to represent the source output so as to save transmission or data storage requirements (while maintaining the necessary fidelity of the data). Obviously, if the source is continuous, then some means of

[1] In a digital alphabet each symbol consists of b bits ($b \geq 1$) and the b-bit symbols can be represented by the integers $\{0, 1, \ldots, 2^b - 1\}$. Hence, a 2^b-*ary* alphabet.

discretising/quantising (e.g., some form of A/D conversion) is generally necessary prior to data compression. An example of a modern data compression technique is VQ described in Chapter 5. Source encoding is usually followed by *encryption* [23] for making the information sequence **u** unreadable by unauthorised recipients. However, this function is omitted from the block diagram in Fig. 2.1.

The *channel encoder* maps the information sequence **u** into a sequence of code symbols (usually binary) called the *codeword* **v** that has the properties of error-correction at the receiving end. The approach is to introduce *redundancy* (excess information) in the channel encoder and to use this redundancy at the channel decoder to reconstruct the information sequence **u** as accurately as possible. This process of adding redundancy to minimise the effects of channel distortions is referred to as *channel coding*.

The modulator maps the code symbols into finite-energy signals suitable for transmission over the *physical channel*, and which are time-superimposed to form the *transmitted signal* $s(t)$. The code symbols may be interfaced to the modulator in various ways [24]. Early applications typically involved binary coding/modulation, in which each binary code symbol is presented to a binary modulator, say a BPSK (binary phase shift keying) modulator, and the modulator produces one of two finite-energy signals. Alternatively, the outputs of the encoder can be subdivided into b-bit blocks, each block mapped into a symbol from a M-ary alphabet $\{0, 1, \ldots, M-1\}$, and the modulator produces one of $M = 2^b$ distinct finite-energy signals to transmit these new symbols unambiguously. The latter technique is particularly useful for bandwidth-limited channels. In general, the signals may represent voltages, currents, electric field density, and so on. There are many possible types of modulation formats, differing in the manner the signals are manipulated. Such manipulations include changing the amplitude, frequency, phase, or both phase and amplitude (QAM) of the transmitted signal $s(t)$, the polarisation of the electromagnetic radiation (in magnetic recording systems), or the pulse position within a modulation interval. The selection of the modulation format largely depends on the limitations of the physical channel (e.g., bandwidth and/or power limitations) [25]. Combined coding and modulation techniques are also possible. Such techniques known as *trellis-coded modulation* (TCM) [26] are particularly well suited for digital transmission over some bandwidth-limited channels (e.g., voice-band telephone channels).

The transmitted signal $s(t)$ enters the physical channel through which communication

takes place. Examples of transmission channels include satellite links, high-frequency (HF) radio links, microwave links, telemetry links, and data storage media. *Noise* $n(t)$ is present in all types of physical channels, and hence the characteristics of the transmitted signal $s(t)$ are altered. Therefore, the *received signal* $\mathbf{r}(t)$ at the output of the analogue channel is a function of the transmitted signal $s(t)$ and the noise signal $n(t)$, that is, $r(t) = s(t) + n(t)$. The noise signal $n(t)$ originates from a number of sources including:

- The addition of noise by electronic equipment used to perform the communication processes, such as filters and amplifiers.
- The influence of external noise sources such as cosmic and atmospheric electromagnetic noise, fading, multipath, man-made noise (e.g., high-voltage power-line corona discharge), intentional jamming by an unfriendly party, etc.
- Distortions due to physical channel limitations, such as the bandwidth limitation of voice-band telephone channels causing intersymbol interference (ISI).

In each case, a well-defined mathematical model exists for modelling the channel noise. Examples of channel noise models are the Gaussian, Rician and Rayleigh channel noise models. The first is commonly referred to as the additive white Gaussian noise (AWGN) channel.

The demodulator processes the received signal $r(t)$ to generate the *received sequence* $\mathbf{r}$, which in turn is passed on to the *channel decoder* for error-correction processing. The received sequence $\mathbf{r}$ could be continuous or discrete, in the latter case (in practice the most common) the symbols are selected from a Q-ary alphabet $\{0, 1, \ldots, Q-1\}$. In addition, if the channel acts on each input symbol independently, then the channel is said to be *memoryless*. In this case, the combination of the M-ary modulator, the transmission link, and the Q-ary demodulator is commonly referred to as the *discrete memoryless channel* (DMC) [27]. A DMC is specified by a set of *transition probabilities* $P(\beta|\alpha)$, $0 \le \beta \le M-1$, $0 \le \alpha \le Q-1$, where α represents a modulator input symbol, β represents a demodulator output symbol, and $P(\beta|\alpha)$ is the conditional probability of receiving β given that α was transmitted. When $Q = M$ the demodulator is said to make *hard-decisions* (e.g., the binary symmetric channel (BSC), where $M = 2$ and $Q = 2$). However, when $Q > M$ or the input to the channel decoder is analogue, the demodulator is said to perform *soft-decisions* (e.g., the binary-input ($M = 2$) AWGN channel with

finite-output ($Q > 2$) quantisation). In this case the channel decoder takes advantage of the extra likelihood information provided to produce the *estimated sequence* $\hat{\mathbf{u}}$ with more certainty. Ideally, the estimated sequence $\hat{\mathbf{u}}$ would be an exact replica of the information sequence $\mathbf{u}$, although this depends upon the level of the channel noise and the error-correcting capability of the channel decoder. Finally, the estimated sequence $\hat{\mathbf{u}}$ is transformed by the source decoder into an estimate of the information source that is delivered to the *information sink* (destination). This usually requires some form of digital-to-analogue (D/A) conversion.

2.2 Error-Control Techniques

The block diagram shown in Fig. 2.1 represents a one-way system since transmission is strictly from the transmitter to the receiver only. Error-control for a one-way system is achieved using *forward error correction* (FEC) [28]. In a FEC system, the channel decoder makes an estimate of the information sequence $\mathbf{u}$ based on hard- or soft-decision inputs without the aid of a communication link back to the transmitter. The *throughput efficiency*[2] of a FEC system is moderate but constant regardless of the channel error rate. A disadvantage of the FEC approach is that its reliability tends to depend on the channel error rate and for high system reliability a powerful code is required. However, this increases the decoding time delay and the implementation complexity of the channel decoder. FEC has numerous applications including satellite communications, digital mobile radio systems and digital magnetic recording systems.

In certain cases, transmission in a communications system can be two-way. In other words, information can be sent in both directions and the transmitter (and receiver) acts like a transceiver. Error control for a two-way system is achieved using *automatic repeat request* (ARQ) techniques [29]. In an ARQ system, the receiver decoder only detects errors and with the aid of a communication link back to the transmitting terminal requests retransmission of the incorrectly received information frames. Since the code employed has no error-correcting capability the encoder/decoder implementation is quite simple. The main advantage with an ARQ system is its inherent high reliability, which is relatively insensitive to the channel error rate. However, the throughput efficiency of an ARQ system is variable and depends upon a number of factors. These factors include the

[2] Throughput efficiency is defined as the ratio of the total number of decoded symbols delivered by the decoder per unit of time to the total number of information symbols transmitted [21].

channel error rate, the frame length and the type of ARQ protocol used. The three fundamental types of ARQ protocol are the stop-and-wait ARQ, the go-back-N ARQ, and the selective-repeat ARQ [21]. ARQ systems are good candidates for providing reliable communication in certain applications, especially those where channel errors occur in bursts. They are mostly employed in data communication systems, such as packet switching data networks and computer communication networks.

A combination of FEC and ARQ techniques forms a hybrid system. A hybrid FEC-ARQ system consists of an FEC system contained within an ARQ system. Such systems combine the advantages of both techniques. The inner FEC system corrects many channel errors, reduces the frequency of retransmissions, and improves the system throughput efficiency. The outer ARQ system provides a very low probability of undetected errors, and increases the system reliability. Therefore, a good hybrid FEC-ARQ system provides higher reliability than a FEC system alone and also higher throughput efficiency than an ARQ system alone. Hybrid FEC-ARQ techniques are used for error control in certain types of data communication systems, such as satellite data links.

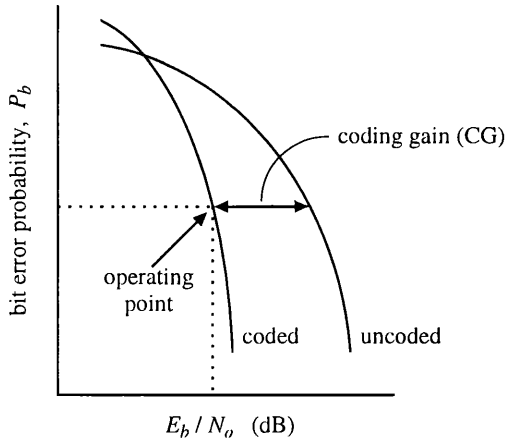


Figure 2.2 Defining CG.

2.2.1 Coding Gain

In practice, the improvement achieved by FEC over uncoded signalling is usually expressed in terms of *coding gain* (CG), as defined in Fig. 2.2. Here, E_b is the mean received energy per information bit and N_o is the single-sided received noise power spectral density. The ratio E_b/N_o (expressed in decibels, dB) is a signal-to-noise (S/N) measure that acts as a figure of merit for various combinations of coding/modulation

schemes. The usual method of calculating CG (see Fig. 2.2) is to plot the *bit error probability*[3] P_b, versus E_b/N_o for both coded and uncoded systems and read the difference in required E_b/N_o at some specified operating P_b (typically 10^{-5}). Such savings in E_b/N_o over uncoded signalling allow either a reduction of the transmitter power/antenna gain or an increase in the bit rate. Sometimes for high P_b, CG can become negative, in which case there is no advantage in using FEC. Note that the use of error-correction increases the channel bandwidth requirement since redundancy is introduced. This penalty results in an increased channel error rate, compared with an uncoded system. Hence, a powerful error-correcting code must be employed not only to provide some reasonable amount of CG but also to compensate for the increased channel bandwidth. It should be noted that CG cannot increase indefinitely but is upper bounded by [30]

$$\mathrm{CG} \leq 10\log(Rd) \tag{2-1}$$

where R is the *code rate* and d is a distance measure of the code. These terms are defined later in this chapter.

2.2.2 Types of Codes for Forward Error Correction

FEC codes can be classified as *block* or *convolutional* codes [20], [21]. With block codes the encoder is presented with an information block of k symbols selected from a w-ary alphabet $\{0, 1, \ldots, w-1\}$, and produces a codeword $\mathbf{v}$ of length n, the code symbols taken from a q-ary alphabet $\{0, 1, \ldots, q-1\}$. For uniqueness of the encoding $w^k \leq q^n$ and since redundancy is introduced $k < n$. Each n-symbol block depends *only* upon a specific k-symbol block and on no others. Therefore, the block encoder has no memory and is implemented by a combinatorial logic circuit. Normally, the input and output alphabets of the encoder are of the same size (the common alphabet size is q), and the alphabet relation is often binary ($q = 2$). The set of q^k codewords of length n produced by such an encoding relation is called a q-ary (n, k) block code. The ratio $R = k/n$ is called the *code rate* and is a measure of the amount of redundancy introduced by the encoder. Typical values in practice for k range from 3 to several hundred and for R from 1/4 to 8/9.

[3] This may be defined as the expected number of bit errors in a given received sequence of bits normalised by the total number of bits in the sequence [27].

Block codes may be classified as *linear* or *nonlinear*. Linear codes are defined by a linear mapping from the space of information blocks to the space of codewords. In addition, they have the important property that two codewords can be added using a suitable definition for addition [i.e., over a *Galois field* GF(q)] to produce a third codeword [20]. The remaining codes are nonlinear codes but these are not particularly important in practical applications of block coding. In general, the selection of a block code for a particular application is based on its error-correction and error-detection capabilities. These capabilities are in turn related to the *minimum distance* $d_{\min}$ of the code, that is, the minimum Hamming distance between any two codewords. For hard-decision decoding, a block code with a given $d_{\min}$ is guaranteed to correct up to $t = \lfloor d_{\min} - 1/2 \rfloor$ channel errors or detect up to $e = d_{\min} - 1$ channel errors per codeword [21], where $\lfloor x \rfloor$ is the integer part of x.

The most important class of linear block codes encountered in practice are cyclic codes [21]. These codes exhibit the property that a cyclic shift of the symbols in a codeword generates a new codeword. As a result of the cyclic property, a number of efficient practical decoding algorithms have been devised for cyclic codes that enable the use of long block codes with a large number of codewords. A major sub-class of cyclic codes with very powerful error-correcting properties is that of Bose-Chaudhuri-Hocquenghem (BCH) codes. The most common type called primitive BCH codes have the following parameters:

$$\begin{aligned} & n = 2^m - 1 \qquad m = 3, 4, \ldots \\ & n - k \le mt \\ & d_{\min} \ge 2t + 1 \end{aligned} \tag{2-2}$$

Reed-Solomon (RS) codes [31] are a class of nonbinary BCH codes that achieve the largest possible $d_{\min}$ for any linear block code with given (n, k) parameters[4]. A t-error-correcting RS code with symbols from a 2^m-ary alphabet has $n = 2^m - 1$ and $k = 2^m - 1 - 2t$, where $m \ge 2$. RS codes are particularly useful for burst error-correction and for use as outer codes in concatenated coding systems (see Section 2.2.3). They are largely employed in optical and digital magnetic recording systems. Other important classes of linear block codes that are cyclic or related to cyclic codes include the Golay,

[4] For nonbinary codes, the distance between two codewords is defined as the number of nonbinary symbols by which they differ.

Hamming, and Reed-Müller codes [20], [21]. Error-detecting codes used in practical communication systems are mainly cyclic-redundancy-check (CRC) codes.

Convolutional codes [21], [32] are a class of FEC codes also referred to as linear *trellis codes*. A convolutional encoder also accepts (small) information blocks of k symbols and produces encoded symbols in small blocks of n symbols (the sequence of n-symbol blocks forms the codeword $\mathbf{v}$). As with block codes, the input and output symbols are usually members of the same q-ary alphabet $\{0, 1, \ldots, q-1\}$ (and often binary), with $n > k$, thus inducing redundancy. However, unlike block codes each block of n symbols depends not only on the k input symbols at that time, but also on several preceding input blocks. Therefore, the convolutional encoder exhibits memory and is implemented by a sequential logic circuit. The rate of convolutional codes is also $R = k/n$. Here, practical values for k range from 1 to 8 and for R from $1/4$ to $7/8$.

Convolutional codes are error-correcting codes, but the error-correcting capability of a convolutional code is somewhat more difficult to define. It generally depends on the distance properties of the code (see Section 2.3.3) and the decoding algorithm employed. Typical applications of convolutional codes (see Section 2.4) are numerous and include satellite communications, spread spectrum systems, mobile communications, military communications and digital modems.

2.2.3 Concatenated Codes

A practical technique for constructing a long code with very powerful error-correcting capability is the combination of a number of shorter codes. Such a code that employs multiple levels of coding is called a *concatenated code* [33]. The main practical advantage of a concatenated code over a single long code is that for the same low error rates, decoder implementation complexity is considerably less than that which would be required by a single long code (block or convolutional). The commonly used approach for generating a concatenated code employs two levels of coding as shown in Fig. 2.3. Normally the outer code is a nonbinary code (e.g., an RS code) while the inner code is a randomly selected binary code (block or convolutional). Such systems can achieve arbitrarily large values of CG. For example, on the AWGN channel with BPSK or QPSK modulation up to 7.5dB of CG at $P_b = 10^{-5}$ can be achieved by the combination of an outer RS code with an inner soft Viterbi-decoded convolutional code [20]. Concatenated

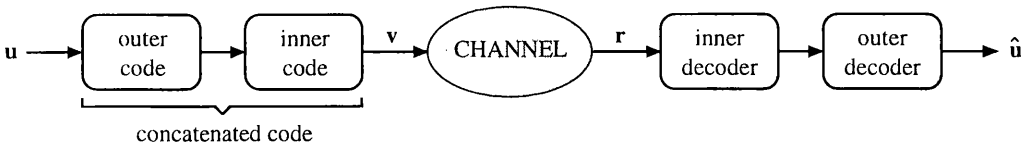


Figure 2.3 Block diagram of a concatenated coding system.

coding schemes are usually employed where very high levels of error-correction are required. Commercial applications include the compact disc (CD) digital audio system [34] (both outer $(32, 28)$ and inner $(28, 24)$ codes are RS codes) and the digital video broadcasting over satellite (DVB-S) [35] system (inner code is a $(204, 188)$ RS code and outer code is a soft Viterbi-decoded $(2, 1, 7)$ convolutional code). In the above applications the nonbinary symbols of the RS codes are *bytes* (1 byte = 8 bits).

2.2.4 Coding Techniques for Burst Noise Channels

In many practical channels, the mapping from a channel input symbol to a channel output symbol is not always an independent event and thus, the channel is said to exhibit memory [27]. Physical origins for this memory are totally varied and may be due to multipath fading[5] (e.g., Rayleigh or Rician fading) on a radio link, occasional burst noise due to hostile jamming, magnetic or optical disk recordings with surface defects, and ISI effects due to channel time dispersion. All of these time-correlated impairments result in statistical dependence among successive symbol transmissions. The result is that channel errors are no longer random but they tend to occur in short bursts.

One method of achieving reliable communication in a burst noise channel is to use an ARQ approach or even better a hybrid FEC-ARQ technique. However, these techniques require the availability of a reverse link, something that is impossible for many practical applications such as digital magnetic recording systems.

Another approach is to use FEC codes specifically designed to correct bursts of errors. Good burst error-correcting codes include the RS codes and the Fire codes from the family of linear block codes, and the Berlekamp-Preparata codes and the Iwadare-Massey codes from the family of convolutional codes [21]. Usually, these codes require enough error-free guard space between bursts to guarantee a certain level of performance.

[5] Multipath fading adds memory to the channel by imposing random amplitude and phase variations onto the transmitted signal.

However, often the channel memory statistics are time varying and thus, it is very difficult to find an accurate model that characterises them [27]. Since each burst error-correcting code is matched to a particular set of channel memory statistics, in practice these techniques are not effective at all times. In addition, burst error-correcting codes do not readily provide for incorporation of soft-decisions, or channel state information, if available. Therefore, burst error-correcting codes have seen relatively little practical application, with the possible exemption of magnetic disks and tape units.

The best technique for combating burst errors especially when soft-decisions are available is to change the ordering of the code symbols before sending them to the channel. This reordering procedure to achieve time-diversity is called *interleaving* [20]. Separating the code symbols in time effectively transforms a channel with memory to a memoryless channel and thus, it allows a powerful random error-correcting code to be used. This technique requires nothing more than an approximation of the duration of the channel memory and hence, it is very resistant to changes in channel memory statistics [27].

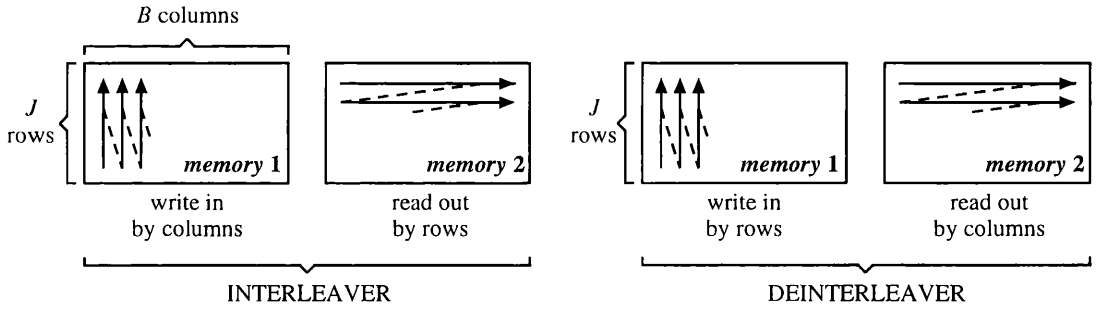


Figure 2.4 The (B, J) block interleaving approach.

Interleaving can be performed in a number of ways. Conceptually, the simplest is *block interleaving* [20] shown in Fig. 2.4. With this approach the interleaver memory space is viewed as a rectangular B-column-by-J-row $(B \times J)$ array of storage locations in which code symbols are written in by columns, and when the memory array is full, they are read out by rows. The deinterleaver performs the inverse operation by writing the received channel output symbols into the rows of a similar array and reading them out by columns after the array is full. For continuous operation, the interleaver and deinterleaver each consist of two such arrays, one for writing in symbols and one for reading out symbols. An interleaving system such as the one illustrated in Fig. 2.4 is called a (B, J) block interleaving system. Choice of the block interleaver parameters depends upon the desired

performance. In general, the (B, J) interleaving approach guarantees that any b channel symbols $2 \leq b \leq B$, in a burst of length b to B are separated by at least J symbols out of the deinterleaver. The major drawbacks of this approach are the double buffering required, the long *throughput delay*[6] ($2BJ$ channel symbol time intervals), as well as the complex deinterleaver synchronisation because of the large number of memory arrays (BJ).

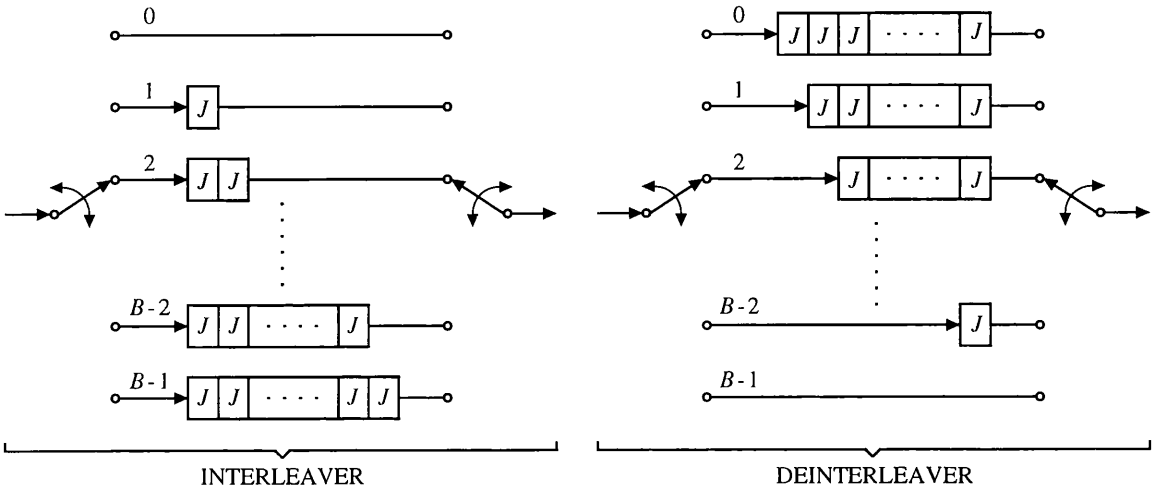


Figure 2.5 The (B, J) convolutional interleaving approach.

A more efficient interleaving scheme is performed by the (B, J) *convolutional interleaving* system [36] shown in Fig. 2.5. The code symbols are shifted sequentially into the bank of B shift registers with increasing lengths. The commutators on the input and output of the interleaver operate synchronously, and after each code symbol is shifted in they move to the next branch. Hence, when a new code symbol enters a shift register the oldest symbol in that register is shifted out to the channel. The deinterleaver performs the inverse operation, and for proper operation the deinterleaver commutators must be synchronised to those of the interleaver. When the commutators are synchronised, there is a constant throughput delay of $BJ(B-1)$ channel symbol time intervals. The performance of a convolutional interleaver is very similar to that of a block interleaver when its parameters are selected in the same manner. Advantages of convolutional interleaving over block interleaving are the reduction in throughput delay and storage requirement, and the somewhat simpler deinterleaver synchronisation.

---

[6] The throughput delay of an interleaving system is defined as the delay from the time a code symbol is written into the interleaver until it is read out of the deinterleaver with a zero channel delay.

2.3 Structures of Convolutional Codes

In this section the fundamental structures and basic properties of convolutional codes are described. The various mathematical methods for describing convolutional codes are first presented followed by the different graphical representations and key properties of convolutional codes.

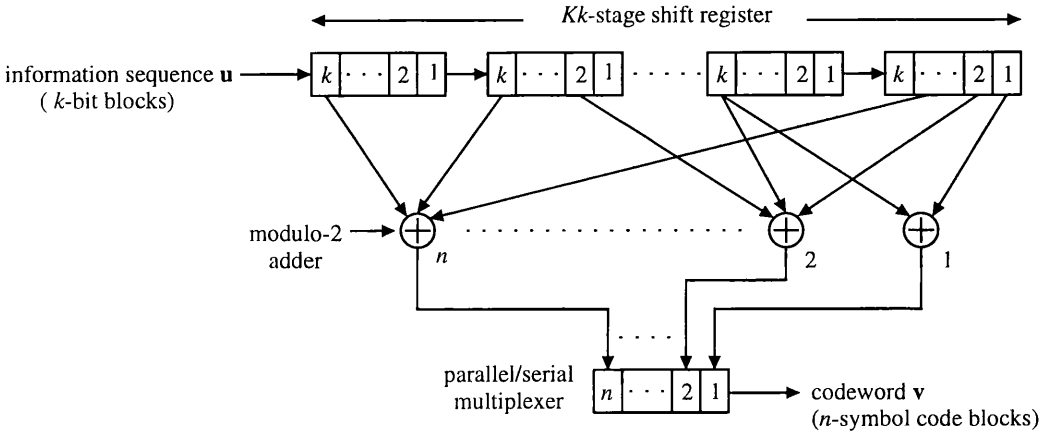


Figure 2.6 General encoder diagram for a (n, k, K) convolutional code.

2.3.1 Encoding of Convolutional Codes

A binary convolutional code is generated by passing the information sequence **u** through a linear finite-state shift machine consisting of a K (k-bit) stages shift register, n modulo-2 adders, and a parallel-to-series multiplexer for serialising encoder outputs into a single codeword **v** (see Fig. 2.6). At each time instant, a k-bit information block enters the shift register and the contents of the last k stages of the shift register are shifted out. The K (k-bit) stages of the register contain the present k-bit block and the previous $(K-1)$ k-bit blocks. The parameter K is known as the *constraint length*[7] and determines the memory order of the convolutional encoder. That is, the number of n-symbol code blocks that are influenced by a k-bit input block. After the current k-bit block has entered the shift register, n linear combinations of the contents of the shift register are computed to generate n code symbols. Over the years, it has become conventional to refer to a convolutional encoder with k inputs, n outputs, and memory order K as a (n, k, K) convolutional encoder. Typically, n and k are small integers ($n > k$), but K must be made

[7] Sometimes the constraint length is defined to be the number of $v = K - 1$ previous k-bit input blocks.

large to achieve low error probabilities. The code rate $R = k/n$ is usually expressed in bits per code symbol [27].

When the information sequence **u** has a finite length of kL bits, a convolutional code can be viewed as a long linear block code [21]. In this case, the corresponding codeword **v** has length $n(L+K-1)$ code symbols (it is assumed that the information sequence **u** is terminated with all "0" blocks to return the encoder to the initial condition) and so, the *effective rate of information transmission* is given by

$$R_T = \frac{kL}{n(L+K-1)} = R\left(1 - \frac{K-1}{L+K-1}\right) \tag{2-3}$$

If $L >> K$, then $L/(L+K-1) \approx 1$, and R_T is approximately equal to R. If L were small, however, R_T would be reduced below R by a fractional amount $(K-1)/(L+K-1)$ called the *fractional rate loss* [21]. Hence, to keep the fractional rate loss small, L must always be much larger than K.

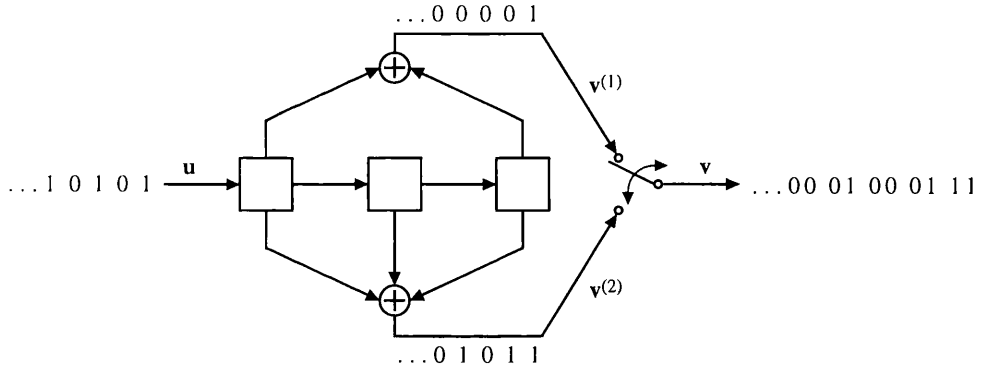


Figure 2.7 Binary convolutional encoder with $k = 1$, $n = 2$, and $K = 3$.

Figure 2.7 gives the block diagram of a very simple binary $(2, 1, 3)$ convolutional encoder that illustrates many of the fundamental concepts of convolutional encoding. The two *output sequences* $\mathbf{v}^{(1)} = (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, \ldots.)$ and $\mathbf{v}^{(2)} = (v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, \ldots.)$ are multiplexed into the codeword $\mathbf{v} = (v_0^{(1)}v_0^{(2)}, v_1^{(1)}v_1^{(2)}, v_2^{(1)}v_2^{(2)}, \ldots)$ at twice the rate of the information sequence $\mathbf{u} = (u_0, u_1, u_2, \ldots)$. Each output sequence $\mathbf{v}^{(j)}$ can be expressed as the convolution of the information sequence **u** with the encoder *impulse response* (or *generator sequence* [21]) $\mathbf{g}^{(l)} = (g_0^{(l)}, g_1^{(l)}, \ldots, g_{K-1}^{(l)})$ specified by the connections between the shift register and the n modulo-2 adders. That is,

$$\mathbf{v}^{(l)} = \mathbf{u} * \mathbf{g}^{(l)}, \qquad l = 1, 2, \ldots, n \tag{2-4}$$

where all operations are in *modulo-2* arithmetic[8] and the discrete convolution $*$ implies that

$$v_i^{(l)} = \sum_{y=0}^{K-1} u_{i-y} g_y^{(l)} = u_i g_0^{(l)} \oplus u_{i-1} g_1^{(l)} \oplus \ldots \oplus u_{i-K+1} g_{K-1}^{(l)} \tag{2-5}$$

The generator sequences for the encoder in Fig. 2.7 are $\mathbf{g}^{(1)} = (1\ 0\ 1)$ and $\mathbf{g}^{(2)} = (1\ 1\ 1)$. Hence, for the information sequence $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$ the two encoding equations are

$$\mathbf{v}^{(1)} = \mathbf{u} * \mathbf{g}^{(1)} = (1\ 0\ 1\ 0\ 1) * (1\ 0\ 1) = (1\ 0\ 0\ 0\ 0\ 0\ 1)$$

$$\mathbf{v}^{(2)} = \mathbf{u} * \mathbf{g}^{(2)} = (1\ 0\ 1\ 0\ 1) * (1\ 1\ 1) = (1\ 1\ 0\ 1\ 0\ 1\ 1)$$

and the resulting codeword after interleaving is $\mathbf{v} = (1\,1, 0\,1, 0\,0, 0\,1, 0\,0)$. Note that in the above calculations and the ones that will follow it is assumed that the encoder was initially filled with all "0" bits and that the information sequence $\mathbf{u}$ is terminated with $n(K-1)$ "0" bits ($=4$ here) to clear the encoder.

If the two generator sequences $\mathbf{g}^{(1)}$ and $\mathbf{g}^{(2)}$ are interleaved and arranged in the matrix

$$\mathbf{G} = \begin{bmatrix} g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_1^{(1)} g_1^{(2)} & & & & \\ & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & & & \\ & & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & & \\ & & & \ldots & & \ldots & \\ & & & & \ldots & & \ldots \end{bmatrix} \tag{2-6}$$

where the blank areas are all "0"s, then the codeword $\mathbf{v}$ corresponding to an information sequence $\mathbf{u}$ can be obtained through matrix multiplication by

$$\mathbf{v} = \mathbf{uG} \tag{2-7}$$

Thus, for the encoder in Fig. 2.7 and the information sequence $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$, the resulting codeword $\mathbf{v}$ obtained using Eq. (2-7) is

[8] In modulo-2 arithmetic addition "$\oplus$" is performed according to the logical XOR function and multiplication "$\cdot$" is performed according to the logical AND function.

$$\mathbf{v} = (1\ 0\ 1\ 0\ 1)\cdot\begin{bmatrix} 11 & 01 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 01 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 01 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 01 & 11 \end{bmatrix}$$

$$= (11, 01, 00, 01, 00, 01, 11)$$

Finally, the convolutional encoding process can be expressed in polynomial form in terms of a unit delay element, D. In this case,

$$\begin{aligned} &\mathbf{u} = (u_0, u_1, u_2, \ldots) \quad \Leftrightarrow \quad \mathbf{u}(D) = u_0 \oplus u_1 D \oplus u_2 D^2 \oplus \ldots \\ &\mathbf{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, v_2^{(j)}, \ldots) \Leftrightarrow \mathbf{v}^{(j)} D = v_0^{(j)} \oplus v_1^{(j)} D \oplus v_2^{(j)} D^2 \ldots \\ &\mathbf{g}^{(j)} = (g_0^{(j)}, g_1^{(j)}, \ldots, g_{K-1}^{(j)}) \Leftrightarrow \mathbf{g}^{(j)} D = g_0^{(j)} \oplus g_1^{(j)} D \oplus \ldots \oplus g_{K-1}^{(j)} D^{K-1} \end{aligned} \tag{2-8}$$

and therefore the encoding equations can be represented by

$$\mathbf{v}^{(j)}(D) = \mathbf{u}(D)\mathbf{g}^{(j)}(D), \qquad j = 1, 2, \ldots, n \tag{2-9}$$

Hence, for the encoder in Fig. 2.7 and using the same information sequence $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$ as earlier, the encoding equations expressed in polynomial form are

$$\begin{aligned} \mathbf{v}^{(1)}(D) &= \mathbf{u}(D) \cdot \mathbf{g}^{(1)}(D) \\ &= (1 \oplus D^2 \oplus D^4) \cdot (1 \oplus D^2) = 1 \oplus D^6 \end{aligned}$$

and

$$\begin{aligned} \mathbf{v}^{(2)}(D) &= \mathbf{u}(D) \cdot \mathbf{g}^{(2)}(D) \\ &= (1 \oplus D^2 \oplus D^4) \cdot (1 \oplus D \oplus D^2) = 1 \oplus D \oplus D^3 \oplus D^5 \oplus D^6 \end{aligned}$$

After multiplexing the resulting codeword polynomial becomes

$$\begin{aligned} \mathbf{v}(D) &= \mathbf{v}^{(1)}(D^2) \oplus D\mathbf{v}^{(2)}(D^2) \\ &= 1 \oplus D \oplus D^3 \oplus D^7 \oplus D^{11} \oplus D^{12} \oplus D^{13} \end{aligned}$$

which corresponds to the same codeword $\mathbf{v} = (11, 01, 00, 01, 00)$ obtained earlier.

2.3.2 Graphical Representation of Convolutional Codes

Three different but related graphical representations [32] have been devised for the study of convolutional encoding, namely, the *state diagram*, the *tree diagram*, and the *trellis diagram*. The encoder for a binary (n, k, K) convolutional code has $S = 2^{(K-1)k}$ different states S_i, $i = 0, 1, \ldots, S-1$, which are defined by the contents of the last $k(K-1)$ register stages. Knowledge of the present state together with knowledge of the next k-bit input block is necessary and sufficient to determine the n-symbol code block. The encoder of Fig. 2.7 has a 3-stage shift register. The first stage represents the present input bit and the latter stages represent the two previous input bits. The states of the encoder as defined before are: $S_0 = 00$, $S_1 = 10$, $S_2 = 01$, or $S_3 = 11$, and are shown in the state diagram in Fig. 2.8. The branch (transition) leaving the present state and leading to the next state indicates the state transition. There are $V = 2^k$ $(= 2$ here) branches entering and leaving from each state and each branch is labelled with the k input bits causing the transition and the corresponding n code symbols. Assuming that the encoder is initially in state S_0, the codeword $\mathbf{v}$ corresponding to any information sequence $\mathbf{u}$ can be obtained by following the path through the state diagram determined by $\mathbf{u}$ and noting the corresponding output code symbols on the branch labels. Therefore, for the encoder in Fig. 2.7 and the information sequence $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$, the corresponding codeword is $\mathbf{v} = (11, 01, 00, 01, 00)$.

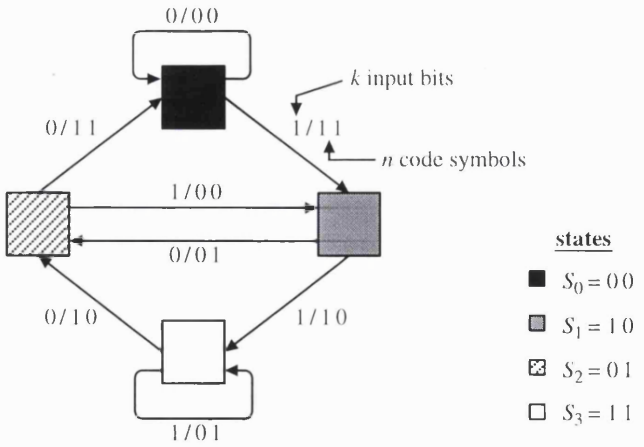


Figure 2.8 State diagram for the encoder in Fig. 2.7.

The tree diagram adds the dimension of time to the state diagram. Each node in the tree diagram corresponds to an encoder state and usually the first node of the tree on the left corresponds to the all-zero state, S_0. As with the state diagram, each tree branch is

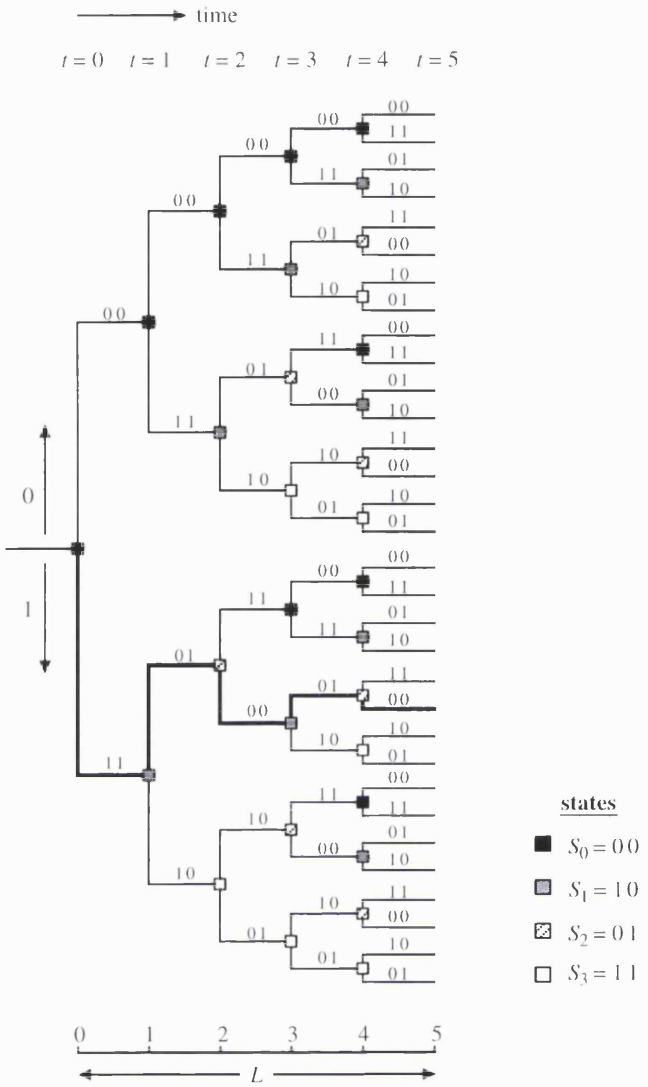


Figure 2.9 Tree diagram for the encoder in Fig. 2.7. Heavy line indicates route for message $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$.

labelled with the k input bits causing the transition and the corresponding n code symbols.

For an information sequence $\mathbf{u}$ of length kL bits there are 2^{kL} possible different codewords of length $n(L+K-1)$ symbols. Each unique codeword is represented by a unique path of length L through the tree diagram, corresponding to L time intervals. Fig. 2.9 depicts the code tree diagram for the encoder in Fig. 2.7. Each node in the tree diagram in Fig. 2.9 corresponds to one of the encoder states $S_0 = 0\,0$, $S_1 = 1\,0$, $S_2 = 0\,1$, and $S_3 = 1\,1$. A path in the tree is traced from left to right according to the information sequence $\mathbf{u}$ that specifies it. In the tree diagram of a $R = 1/n$ convolutional code the upper branch leaving a node corresponds to a "0" input bit, while the lower branch corresponds to a "1" input bit. Assuming that the encoder in Fig. 2.7 is initially in state S_0, then the

heavy line in Fig. 2.9 indicates the path corresponding to the information sequence $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$. This corresponds to the codeword $\mathbf{v} = (11, 01, 00, 01, 00)$. Note that there are 32 possible different paths (or codewords) through this tree diagram over 5 levels. Of course, as the length of the information sequence $\mathbf{u}$ expands, the number of possible paths grows exponentially, and the tree diagram representation of the encoding process becomes impractical.

Referring to the tree representation in Fig. 2.9, it is clear that after the first K ($= 3$ here) levels, the structure repeats itself. In particular, the set of branches emanating from nodes that correspond to the same encoder state are identical. This leads to the trellis diagram [2] representation in which identical branches coming from the same state are merged into a single branch. Fig. 2.10 shows such a diagram for the encoder in Fig. 2.7 starting from state S_0. The first row nodes correspond to the state $S_0 = 00$, the second and subsequent row nodes correspond to the states $S_2 = 01$, $S_1 = 10$, and $S_3 = 11$. Hence the number of nodes per time interval equals the number of states, i.e., $S = 2^{(K-1)k}$ ($= 4$ here). As with the tree diagram of a $R = 1/n$ convolutional code, the upper branch leaving a node corresponds to a "0" input bit while the lower branch corresponds to a "1" input bit, and each branch is labelled with the n code symbols corresponding to the state transition. The heavy line in Fig. 2.10 indicates the path corresponding to the information sequence $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$. This corresponds to the codeword $\mathbf{v} = (11, 01, 00, 01, 00)$, as obtained in the tree diagram case.

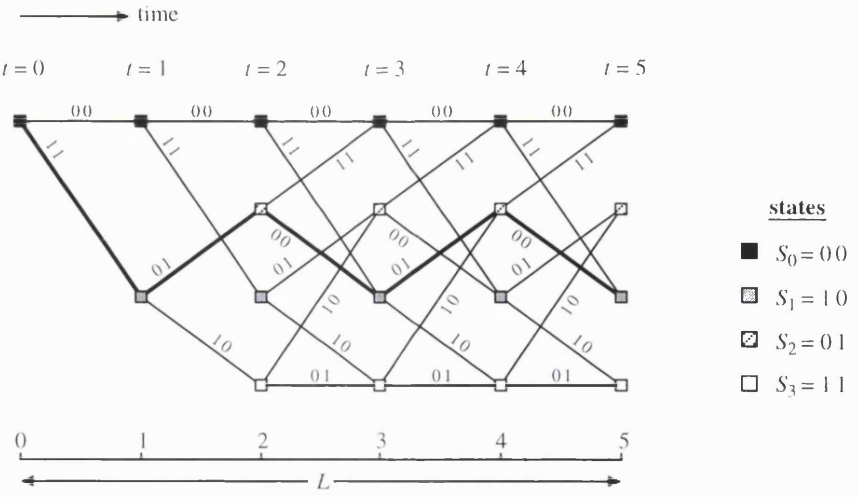


Figure 2.10 Trellis diagram for the encoder in Fig. 2.7. Heavy line indicates route for message $\mathbf{u} = (1\ 0\ 1\ 0\ 1)$.

2.3.3 Distance Properties of Convolutional Codes

As mentioned in Section 2.2.2 the error-correcting capability of a convolutional code depends not only on the distance properties of the code but also on the particular decoding algorithm employed. As for block codes, the Hamming distance structure of a convolutional code is of primary importance for hard-decision decoding. Consider an $R = k/n$ convolutional code and let the information sequence $\mathbf{u}$ and the corresponding codeword $\mathbf{v}$ be truncated at length L such that

$$[\mathbf{u}]_L = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{L-1}) = (u_0^{(1)} u_0^{(2)} \ldots u_0^{(k)}, u_1^{(1)} u_1^{(2)} \ldots u_1^{(k)}, \ldots, u_{L-1}^{(1)} u_{L-1}^{(2)} \ldots u_{L-1}^{(k)})$$

$$[\mathbf{v}]_L = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{u}_{L-1}) = (v_0^{(1)} v_0^{(2)} \ldots v_0^{(n)}, v_1^{(1)} v_1^{(2)} \ldots v_1^{(n)}, \ldots, v_{L-1}^{(1)} v_{L-1}^{(2)} \ldots v_{L-1}^{(n)})$$

The L-th order *column distance function* $d_c(L)$ of a convolutional code is the minimum Hamming distance between all pairs of codewords truncated at length L given that the input information sequences corresponding to the pair of codewords $\mathbf{v}'$ and $\mathbf{v}''$ differ in the first k-bit block. [21]. Thus, $d_c(L)$ is defined as

$$d_c(L) = \min\{d_H([\mathbf{v}']_L, [\mathbf{v}'']_L) \,\big|\, \mathbf{u}_0' \neq \mathbf{u}_0''\} \tag{2-10}$$

Since convolutional codes are linear, the minimum Hamming distance between all pairs of codewords is equal to the minimum Hamming weight of the nonzero codewords and thus, $d_c(L)$ simplifies to

$$d_c(L) = \min\{W_H([\mathbf{v}]_L) \,\big|\, \mathbf{u}_0' \neq 0\} \tag{2-11}$$

where $W_H([\mathbf{v}]_L)$ is the Hamming weight of the codeword $\mathbf{v}$. The column distance function $d_c(L)$ is a nondecreasing function of L, and has two particular values of specific interest: $d_{\min}$, the *minimum distance* of the code when $L = K$; and d_{free}, the *free distance* of the code when $L \to \infty$ [21]. That is,

$$d_{\min} = d_c(K) \tag{2-12}$$

and

$$d_{\text{free}} = \lim_{L \to \infty} d_c(L) \tag{2-13}$$

These two values of $d_c(L)$ define the t-error-correcting capability of the code $t = \lfloor d_c(L) - 1/2 \rfloor$, where $\lfloor x \rfloor$ is the integer part of x. The early decoding techniques

developed for convolutional codes (e.g., *threshold decoding* [37] described in Chapter 3) used only the first nK symbols of the received sequence to decode a k-bit information block. Consequently, for such decoders the error-correcting capability of a convolutional code is a strong function of $d_{\min}$. On the other hand, the VA uses the entire received sequence to decode a k-bit information block. In this case, the error-correcting capability of a convolutional code is a strong function of d_{free}.

Usually, $d_{\min} \leq d_{\text{free}}$ but for many convolutional codes $d_{\text{free}} = d_{\min}$. For convolutional codes with small values of K, d_{free} may be determined from the state and trellis diagrams by considering all the paths that diverge from and remerge with state S_0. For $R = 1/n$ convolutional codes d_{free} is bounded by [24]

$$d_{\text{free}} \leq \min_{L \geq 1} \left\lfloor \frac{2^{L-1}}{2^L - 1}(k + L - 1)n \right\rfloor \tag{2-14}$$

where $\lfloor x \rfloor$ denotes the largest integer contained in x. Convolutional codes optimised for maximum d_{free} can be found in [21]. The complete column distance function $d_c(L)$ of the code of Fig. 2.7 is plotted in Fig. 2.11.

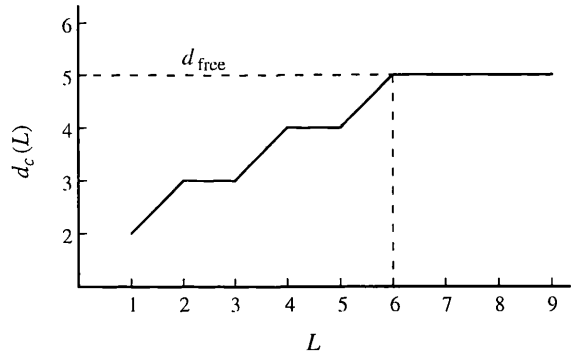


Figure 2.11 Column distance function of the code in Fig. 2.7 ($d_{\text{mim}} = 3$, $d_{\text{free}} = 5$).

2.3.4 Catastrophic Convolutional Codes

Convolutional codes must be selected such that the codewords from the encoder are as different as possible. If the convolutional code is not selected carefully, it can exhibit what is called *catastrophic error-propagation* [21]. Such a code is called a *catastrophic code*. A catastrophic code is defined as a code for which there exist two encoder input sequences that differ in an arbitrarily large number of symbol positions which are encoded into codewords that differ only in a finite number of symbol positions.

Catastrophic encoders have loops in their state diagrams that accumulate zero Hamming weight other than the loop around state S_0. An example of a catastrophic convolutional encoder and its state diagram are shown in Fig. 2.12.

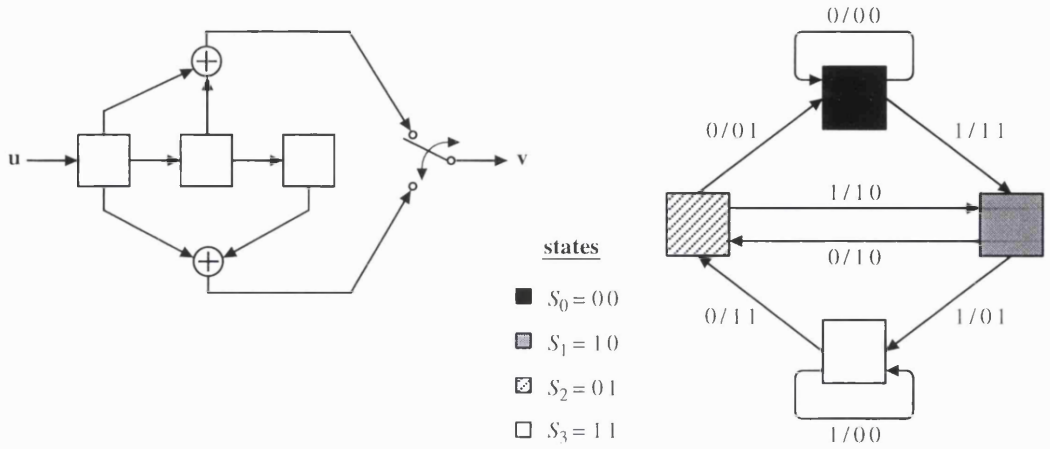


Figure 2.12 A convolutional code exhibiting catastrophic error propagation.

The term catastrophic refers to the potential for a convolutional decoder to be diverted onto an incorrect trellis path by a finite-length span of channel errors, and even with a subsequent error-free channel, remain on this incorrect path indefinitely. The necessary and sufficient condition for a $R = 1/n$ convolutional code to be catastrophic is that its n generator polynomials (see Section 2.3.1) have a common factor (in modulo-2) [38]. Fortunately, for all $R = 1/n$ convolutional codes and a specific constraint length K, only the fraction $1/(2^n - 1)$ are catastrophic [27]. It should be noted that in addition to this type of catastrophic failure caused by bad code design, the decoder structure itself might also cause this phenomenon to happen (see Chapter 3).

2.3.5 Systematic and Nonsystematic Convolutional Codes

An important subclass of convolutional codes that are never catastrophic is the class of *systematic* convolutional codes [21]. In a $R = k/n$ systematic convolutional code, in each n-symbol code block, the first k symbols are exact replicas of the k inputs, and the last $(n-k)$ symbols are parity symbols. Any other convolutional code is said to be *nonsystematic*.

Systematic convolutional codes have the advantage that they allow for easier inversion of the operation of the encoder. However, for a given R and K set, the maximum d_{free}

achievable with systematic codes is smaller than for nonsystematic codes. This is reflected in their error-correcting capabilities. In fact, it can be shown [27] that for asymptotically large K, the error-correcting capability of a systematic code of constraint length K is approximately the same as that for a nonsystematic code of constraint length $K(1-k/n)$, while requiring exactly the same optimal decoder complexity.

2.3.6 Punctured Convolutional Codes

In bandwidth-limited channels high-rate $R=k/n$ convolutional codes are usually employed since redundancy increases the channel bandwidth requirement. The structure of these codes is somewhat complex since there are 2^k branches entering and leaving each state rather than just 2 branches as for the low-rate $R=1/n$ codes. Inevitably, this makes the implementation of their decoders more difficult. A technique that has been proposed to overcome this problem is called *puncturing* [39]. The idea is to periodically delete certain code symbols from a low-rate convolutional code (the parameter k is kept fixed), thereby producing a convolutional code with effectively higher rate. For example, if every fourth code symbol is deleted from the output of the encoder in Fig. 2.7, a *punctured rate* R_p ($=2/3$ here) convolutional code is obtained with trellis as shown in Fig. 2.13. This procedure can be generalised to delete $P-1$ out of $2P$ code symbols from an $R=1/2$ encoder to generate a code of rate $R_p=P/(P+1)$, $P\geq 2$. In general, from a $R=1/n$ parent code the achievable punctured rates are

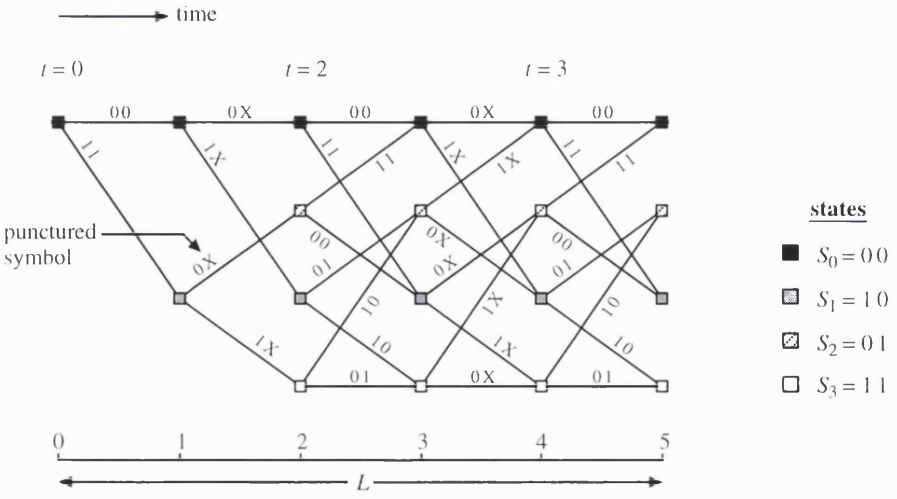


Figure 2.13 Trellis diagram for the encoder in Fig. 2.7 punctured to $R_p = 2/3$.

$$R_p = \frac{P}{P+\gamma}, \qquad \gamma = 1, 2, \ldots, (n-1)P \tag{2-15}$$

The most popular punctured rates in practice are $2/3$, $3/4$, $5/6$ and $7/8$. These are derived from a $R=1/2$ parent code. It should be noted that puncturing does not necessarily introduce weakness in the error-correcting capability of a code since in most cases the optimum punctured code of a given rate attains the maximum d_{free} of the best non-punctured code at that rate [39].

2.4 Applications of Convolutional Codes

In terms of actual applications of coding techniques to date, convolutional codes are perhaps the most popular choice in practice. This is mainly due to the existence of an optimum decoding algorithm (i.e., the VA) that can be easily implemented for either soft- or hard-decisions, thus making convolutional coding a more economical approach in practice than competing block coding techniques for similar levels of error-correction. Typical applications of convolutional codes are satellite communications, mobile communications, voice-band data communications and digital broadcasting systems. To date, the realisation of the coding part of these applications uses a DSP (digital signal processor) or a digital ASIC (application specific integrated circuit) solution. The choice depends mainly on the data rate requirements.

2.4.1 Applications in Satellite Communications

Convolutional codes found their first application in early deep-space communications where the available power from the spacecraft is very restricted. Examples of deep-space missions include the NASA (National American Space Agency) Pioneer, Voyager and Gallileo missions [21]. A few years later, error-control convolutional coding was applied to digital satellite communication systems not only to overcome power limitation, but also to operate in bandwidth and interference-limited transmission channels [40]. Tables 2.1 and 2.2 summarise the convolutional coding parameters employed in some of INMARSAT (International Maritime Satellite Organisation) and INTELSAT (International Telecommunication Satellite Organisation) services, respectively.

Table 2.1 Convolutional coding parameters employed in INMARSAT.

Standard	Service	R	R_p	K	Data rate (kbit/s)	Realisation
INMARSAT B	Voice	–	3/4	7	16	ASIC/DSP
	Data	1/2	–	7	9.6	
	Shore-mobile telex	1/2	–	7	6	
	Mobile-shore telex	1/2	–	7	24	
INMARSAT-C	Data	1/2	–	7	0.6	DSP
	Shore-mobile telex	1/2	–	7	0.6	
	Mobile-shore telex	1/2	–	7	0.6	
INMARSAT-M	Voice	–	3/4	7	6.4	ASIC/DSP
	Data	1/2	–	7	2.4	
INMARSAT-Aero	Voice	1/2	–	7	2.4-9.6	ASIC/DSP
	Data	1/2	–	7	0.6-10.5	
High-speed data	Mobile-shore	1/2	–	7	56	ASIC

Table 2.2 Convolutional coding parameters employed in INTELSAT.

Service	R	R_p	K	Data rate (kbit/s)	Realisation
INTELSAT-IBS*	1/2	3/4	7	64-8448	ASIC
INTELSAT-IDR†	1/2	3/4	7	64-44736	ASIC

*INTELSAT Business Service, †INTELSAT-Intermediate Data Rate.

2.4.2 Applications in Mobile Communications

In mobile communications, the available power and bandwidth are also very restricted. The transmitted signal is subject to multipath fading, and the size of the mobile transceiver is also restricted. Therefore, recent design and development of digital cellular mobile radio systems make extensive use of error-control coding (convolutional and/or block coding) to enhance the system performance. Examples of cellular radio systems employing convolutional coding (see Table 2.3) include the European GSM (Group Spécial Mobile) system and the North American IS-95 (interim standard 95) and IS-136 (interim standard 136) systems [41]. The GSM and IS-136 are bandwidth-limited systems whereas the IS-95 is a power-limited system.

2.4.3 Applications in Voice-Band Data Communications

Channel coding techniques are not useful for transmission over some bandwidth-limited channels such as the general-switched telephone network (GSTN). In the GSTN the

Table 2.3 Convolutional coding parameters for various digital radio systems.

Standard	Data type	R	R_p	K	Frame size	Frame rate	Realisation
GSM	Voice	1/2	-	5	189 bits	50 Hz	DSP
	Data - 9.6	1/2	57/61	5	244 bits	50 Hz	
	Data - 4.8	1/3	-	5	152 bits	50 Hz	
	Data - 2.4	1/6	-	5	76 bits	50 Hz	
IS-95	Fwd voice	1/2	-	9	192 bits	50 Hz	DSP
	Rev. voice	1/3	-	9	192 bits	50 Hz	
IS-136	Voice	1/2	-	6	89 bits	50 Hz	DSP
	FACCH*	1/4	-	6	65 bits	50 Hz	

*Fast associated control channel.

bandwidth typically extends from 300Hz to 3.4kHz and so conventional error-control coding techniques are not suitable since channel bandwidth expansion here is not an option. However, it is possible to achieve significant values of CG over conventional M-ary modulation techniques without compromising bandwidth efficiency by employing TCM techniques [26]. TCM schemes employ M-ary modulation techniques in combination with trellis coding and on the AWGN channel CG values of more than 6dB over uncoded M-ary modulation are possible. Such improvements are obtained without bandwidth expansion or reduction of the effective information rate as would be required by conventional error-control coding techniques. In TCM trellis codes are optimised for maximum Euclidean free distance, rather than for maximum d_{free}, and redundancy is introduced by expanding the modulation signal alphabet to avoid channel bandwidth expansion.

For over a decade now, TCM schemes have been adopted by the ITU-T (International Telecommunication Union – Telecommunication Standardisation Sector) for use in new generation high-speed voice-band data modems over the GSTN. Examples of such modems include the ITU-T Recommendations V.32, V.32 *bis* and V.34. A summary of their key specification parameters is given in Table 2.4.

Table 2.4 ITU-T standards for high-speed voice-band data modems.

Standard	R	States	Modulation scheme	Data rate (kbit/s)	Realisation
V.32	2/3	8	2-D TMC/32-CR	4.8 or 9.6	DSP
V.32 bis	2/3	8	2-D TMC/128-CR	14.4	DSP
V.34	2/3, 3/4, 4/5	16, 32, 64	4-D TMC/960-QAM	28.8	DSP

2.4.4 Applications in Digital Broadcasting

Sound and television broadcasting are undergoing a revolutionary change from the established analogue to digital standards, where digital signal processing techniques promise vast improvements in quality and variety of services. Two important innovations of the video and audio broadcasting era are the development of the DVB (Digital Video Broadcasting) [35] and DAB (Digital Audio Broadcasting) [42] systems that are currently being implemented around the world. The DVB-S (satellite), the DVB-T (terrestrial) and the DAB systems employ convolutional coding as part of their FEC (see Table 2.5).

Table 2.5 Convolutional coding parameters for digital audio and video broadcasting.

Standard	R	R_p	K	Data rate (Mbit/s)	Realisation
DVB-S	1/2	2/3, 3/4, 5/6, 7/8	7	25-50	ASIC
DVB-T	1/2	2/3, 3/4, 5/6, 7/8	7	Variable	ASIC
DAB	1/2	–	7	2.4	ASIC

Chapter

3

Convolutional Decoding Methods and the Modified Feedback Decoding Algorithm

OVER the years of convolutional coding research, several convolutional decoding approaches have been developed. In 1961 Wozencraft [43] introduced the first practical convolutional decoding technique that was the first of a class of probabilistic search procedures called *sequential decoding*. Although sequential decoding offered near optimum decoding performance, its implementation at that time was very expensive. Therefore, soon after, a group of sub-optimum but relatively simple to implement convolutional decoding techniques called *feedback decoding* [44] were introduced. Feedback decoding was particularly suited to hard-decision applications and offered an inferior coding performance compared to sequential decoding. The third important approach to the decoding of convolutional codes was introduced in 1967 by Viterbi [1]. Viterbi decoding or simply the VA [2], which is in fact a special application of dynamic programming to digital communications [45], provides a practical method for realising *maximum likelihood* (ML) decoding of convolutional codes. Thus, over the last two decades or so, the VA has become the most widely employed convolutional decoding technique in many types of digital communication system. In addition, apart from decoding convolutional codes the VA is considered an attractive solution to a variety of

digital estimation problems [2], and thus has numerous other applications including digital magnetic recording systems and TCM systems.

This chapter describes the three generic convolutional decoding techniques with emphasis on Viterbi and feedback decoding since they form the basis for the development of the MFDA. The chapter begins with a description of the VA as applied to convolutional decoding, followed by implementation issues. Other applications of the VA are then identified and digital magnetic recording is explained. Following that, the two sub-optimum convolutional decoding techniques, namely sequential and feedback decoding, are described. The remaining part of the chapter is devoted to the MFDA.

Figure 3.1 The convolutional decoding problem.

3.1 The Viterbi Algorithm

Figure 3.1 shows a simplified model of a communication system employing a convolutional encoder/decoder and a DMC with an M-ary input alphabet and a Q-ary output alphabet, where $Q \geq M$. The decoder must produce an estimate $\hat{\mathbf{u}}$ of the information sequence $\mathbf{u}$ based on the received sequence $\mathbf{r}$ with minimum probability of error. For ML decoding, the decoder must find that codeword $\mathbf{v}'$ which maximises the *likelihood function* $P(\mathbf{r}|\mathbf{v})$ given that all codewords $\mathbf{v}$ are equally probable [27]. Because of the memoryless channel assumption this may be expressed as

$$P(\mathbf{r}|\mathbf{v}') = \max_{\text{all } \mathbf{v}} P(\mathbf{r}|\mathbf{v}) \tag{3-1}$$

Furthermore, for any $R = k/n$ convolutional code and an information sequence $\mathbf{u}$ of length kL bits, $P(\mathbf{r}|\mathbf{v})$ can be expressed as

$$P(\mathbf{r}|\mathbf{v}) = \prod_{i=0}^{L-1} P(\mathbf{r}_i|\mathbf{v}_i) = \prod_{i=0}^{L-1}\prod_{l=1}^{n} P(r_i^{(l)}|v_i^{(l)}) \tag{3-2}$$

where $\mathbf{r}_i$ is the ith branch of $\mathbf{r}$, $\mathbf{v}_i$ the ith branch of $\mathbf{v}$, $r_i^{(l)}$ the lth symbol of $\mathbf{r}_i$, and $\mathbf{v}_i^{(l)}$ the lth code symbol of $\mathbf{v}_i$. Since logarithms (to any base) are monotonically increasing,

the estimate that maximises $P(\mathbf{r}|\mathbf{v})$ is also the estimate that maximises the *log-likelihood function* $\log P(\mathbf{r}|\mathbf{v})$. Thus the equivalent to Eq. (3-2) is

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_{i=0}^{L-1} \log P(\mathbf{r}_i|\mathbf{v}_i) = \sum_{i=0}^{L-1}\sum_{l=1}^{n} \log P(r_i^{(l)}|v_i^{(l)}) \tag{3-3}$$

The term $\log P(\mathbf{r}|\mathbf{v})$ in Eq. (3-3) is referred to as the *path metric* Γ, whereas the terms $\log P(\mathbf{r}_i|\mathbf{v}_i)$ and $\log P(r_i^{(l)}|v_i^{(l)})$ are referred to as the *branch metric* λ_i and *symbol metric* $\mu_i^{(l)}$, respectively. Thus, finding the *most-likely* path through the trellis or tree diagrams is equivalent to selecting the path with the largest metric. Unfortunately, the number of paths for an information sequence of length kL bits is 2^{kL}, thus this "brute force" ML decoder quickly becomes impractical as L increases.

With the VA, it is possible to greatly minimise the computational effort required for ML decoding by taking advantage of the remerging path structure of the trellis diagram (see Section 2.3.1). Given a received sequence $\mathbf{r}$ corrupted by Gaussian noise, the VA computes the most-likely path (or codeword) through the trellis recursively. At each time instant, the VA compares the metrics of all $V = 2^k$ paths entering each state S_i and *only* stores the path with the largest metric, called the *survivor*, together with its metric. When a tie appears (i.e., the remerging paths have the same metric) the VA arbitrarily eliminates all paths but one. Defining $\Gamma_{j,t}$ as the metric of the survivor at state S_i at time t and $\lambda_{ji,t}$ as the branch metric of the transition from state S_j at time t to state S_i at time $t+1$, then the metric of the optimum path leading to state S_i at time $t+1$ is calculated by the following recursion relationship[1]

$$\forall i: \quad \Gamma_{i,t+1} = \underbrace{\max}_{\forall j} (\Gamma_{i,t} + \lambda_{ji,t}) \qquad i = 0, 1, \ldots, S-1 \tag{3-4}$$

Equation (3-2) is called the *add-compare-select* (ACS) operation [46] and is the essence of the VA. This is the fundamental operation that reduces the search for the ML path from 2^{kL} computations for an information sequence of length kL bits to $Lk2^{k(K-1)}$ computations only. It should be noted that in practice the metrics are realised using distance measures; squared Euclidean distance for soft-decoding (in the case of AWGN) and Hamming distance for hard-decoding [47]. Moreover, in a real-time Viterbi decoder the path metrics are stored in the *metric memory* whereas the history of the survivors at

[1] Note that the *max* function can be converted to a *min* function with an inversion in the sign of the branch metrics.

each state is stored in the (digital) *path memory*. Implementation considerations for the VA are dealt with in Section 3.2.

Once the VA has processed the entire received sequence $\mathbf{r}$, the survivor with the largest metric is selected to be the most-likely path, and the information bits associated with it constitute the estimated sequence $\hat{\mathbf{u}}$. This is achieved by tracing the most-likely path backwards with the aid of the information stored in the path memory. In practice, convolutional codes are used with arbitrarily long information sequences (i.e., $L \to \infty$). Thus, in order to avoid long decoding delays before starting to deliver the estimated sequence $\hat{\mathbf{u}}$ and to reduce the huge path memory that would be required, it is necessary to truncate the survivors to some manageable *decoding depth* W, $1 \le W \le L$. That is, after each time step, the survivor with the largest metric is traced back for W branches, and the k-bit information block corresponding to the branch traced back is the decoded information. Since the survivors tend to remerge a few constraint lengths of branches back, a memory larger than this has little impact on performance. For most $R = 1/2$ convolutional codes setting $W \approx 5K$ is a good choice in the sense that the loss of CG resulting from the truncation is negligible [48]. Slightly smaller W are suitable with lower rate codes and larger path memories are required with higher rate codes [49]. Fig. 3.2 illustrates the VA decoding process for the encoder in Fig. 2.7. These figures illustrate the progression through the trellis diagram for hard-decision Viterbi decoding and a particular received sequence $\mathbf{r}$. Since transmission is over the BSC, minimising the error probability is equivalent to choosing the path that is at minimum Hamming distance from the received sequence $\mathbf{r}$.

Viterbi decoding is particularly useful for efficient communication at very high data rates (typically about 25Mbit/s), where moderate error rates are required and short codes are used (i.e., $K \le 9$). On the BSC and for large E_b/N_o, the bit error performance of Viterbi decoding is approximated by [21]

$$P_e \approx C_v \exp\left(\frac{-Rd_{\text{free}}E_b}{2N_o}\right) \tag{3-5}$$

where C_v is a constant associated with code structure. For a binary-input AWGN channel with unquantised outputs ($Q = \infty$), the approximation (for large E_b/N_o) becomes [21]

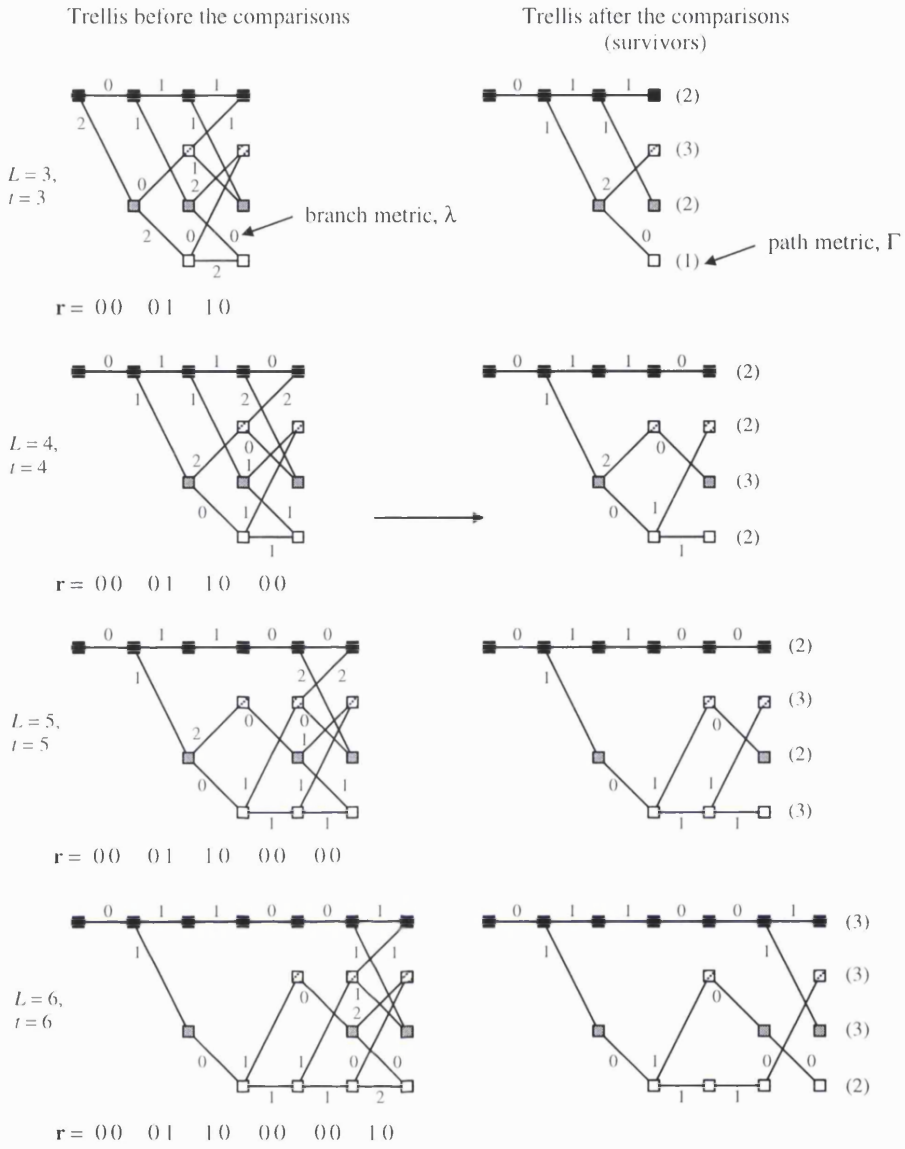


Figure 3.2 Evolution of survivors for hard-decision Viterbi decoding.

$$P_e \approx C_v \exp\left(\frac{-Rd_{\text{free}}E_b}{N_o}\right) \tag{3-6}$$

Comparing Eq. (3-5) with Eq. (3-6) it is observed that the exponent of the latter is larger by a factor of 2, which asymptotically corresponds to a 3dB extra CG (using Eq. 2-1) over hard-decision decoding. For small values of E_b/N_o the improvement reduces to about 2dB and for $Q=8$ there is about 0.25dB loss in CG compared to infinite quantisation [48].

3.2 Considerations for the Implementation of the VA

The functional block diagram of a Viterbi decoder is shown in Fig. 3.3. The decoder is composed of the following blocks [20]: branch metric computer (BMC), add-compare-select (ACS), storage survivor memory (SSM), and output decision (OD). The BMC block performs multiplication, the ACS block performs addition, maximum or minimum selection and path metric storage, the SSM block keeps track of the decisions made in the ACS block in a hypothesised digital representation and the OD block outputs the decoded data. In addition, the decoder is equipped with a synchronisation function [50] so that it can determine the beginning of each n-symbol block in the received sequence $\mathbf{r}$.

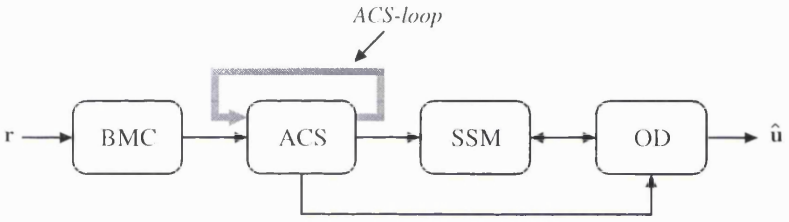


Figure 3.3 Simplified block diagram of a Viterbi decoder.

3.2.1 The BMC Block

Each time a new n-symbol block is received, the BMC block computes a new set of branch metrics, one for each branch of the trellis diagram. In most cases the BMC block is based on a look-up table containing the various symbol metrics. The resulting branch metrics form the inputs to the ACS block. For optimal VA performance in AWGN, the branch metric of a given transition is defined as the squared Euclidean distance between the received n-symbol block, and the n-symbol code block of that transition. That is, the branch metric $\lambda_{ji,t}$ in Eq. (3-4) is given by

$$\lambda_{ji,t} = (\mathbf{r}_t - \mathbf{v}_{ji})^2 = \sum_{l=1}^{n} \mu_{ji}^{(l)} = \sum_{l=1}^{n} (r_t^{(l)} - v_{ji}^{(l)})^2 \tag{3-7}$$

where $\mathbf{r}_t = (r_t^{(1)}, r_t^{(2)}, \ldots, r_t^{(n)})$ is the received block at time t, $\mathbf{v}_{ji} = (r_{ji}^{(1)}, r_{ji}^{(2)}, \ldots, r_{ji}^{(n)})$ is the code block of the transition from state S_j to state S_i, and $\mu_{ji}^{(l)}$ $(1 \le l \le n)$ is the lth symbol metric of the transition from S_j at time t to S_i at time $t+1$. By expanding Eq. (3-7) into

$$\lambda_{ji,t} = \sum_{l=1}^{n} (r_t^{(l)})^2 - 2\sum_{l=1}^{n} (r_t^{(l)} \cdot v_{ji}^{(l)}) + \sum_{l=1}^{n} (v_{ji}^{(l)})^2 \qquad (3\text{-}8)$$

it is observed that the first term on the right of Eq. (3-8) is common to all branch metrics, and thus it can be discarded since it contributes equally to all branch metrics. In fact, the metrics can always be scaled by adding (or subtracting) some constant to all of them or by multiplying (or dividing) all of them by a positive constant.

In the case of antipodal signalling (BPSK) where the normalised code symbols are represented by -1 and $+1$ depending on whether the code symbol is a "0" or a "1", respectively, only negation operations are required to compute the symbol metrics [47] (the last term in Eq. (3-8) is always 1 and so it can be neglected). Thus,

$$\mu_{ji,t}^{(l)} = -r_t^{(l)} \cdot v_{ji}^{(l)} = \begin{cases} -r_t^{(l)} \text{ for } v_{ji}^{(l)} = +1 \\ r_t^{(l)} \text{ for } v_{ji}^{(l)} = -1 \end{cases}, \qquad 1 \le l \le n \qquad (3\text{-}9)$$

Note that in order to retain the term *max* in Eq. (3-4) the sign of the results in Eq. (3-9) must be inverted. Otherwise the term *max* in Eq. (3-4) must be replaced by *min*. Note also that to be able to implement Eq. (3-7) exactly, an analogue realisation must be adopted. However, in practice the received sequence is usually quantised to eight levels before being fed into the decoder and thus, each symbol metric is represented by a 3-bit binary word.

For the BSC with a channel error rate of $p < 1/2$, Hamming distance is the preferred metric. In this case the per symbol metric is given by

$$\mu_{ji,t}^{(l)} = \begin{cases} 0, & r_t^{(l)} = v_{ji}^{(l)} \\ -1, & r_t^{(l)} \neq v_{ji}^{(l)} \end{cases}, \qquad 1 \le l \le n \qquad (3\text{-}10)$$

which may be realised using an XOR circuit. The sign of the results in Eq. (3-10) is in the correct format required to retain the term *max* in Eq. (3-4).

3.2.2 The ACS Block

The ACS block takes the branch metrics computed by the BMC block and computes the path metrics at each node in the trellis. At each time instant, the survivor path for each state is selected, and the SSM block is updated accordingly. The operations performed by

the ACS block are better understood by breaking the trellis up into a number of identical modules as shown for the $R = 1/n$ case in Fig. 3.4a. Since the entire trellis is constructed from replicas of this "butterfly" module, the computational unit shown in Fig. 3.4b can be used recursively to realise the ACS block, and hence Eq. (3-4). Since the basic operations performed by this unit are addition, comparison, and maximum or minimum selection (and storage), the unit is referred to as the *add-compare-select unit* (ACSU). It should be noted that the ACSU in Fig. 3.4b might also be employed in high-rate punctured codes (see Section 2.3.6).

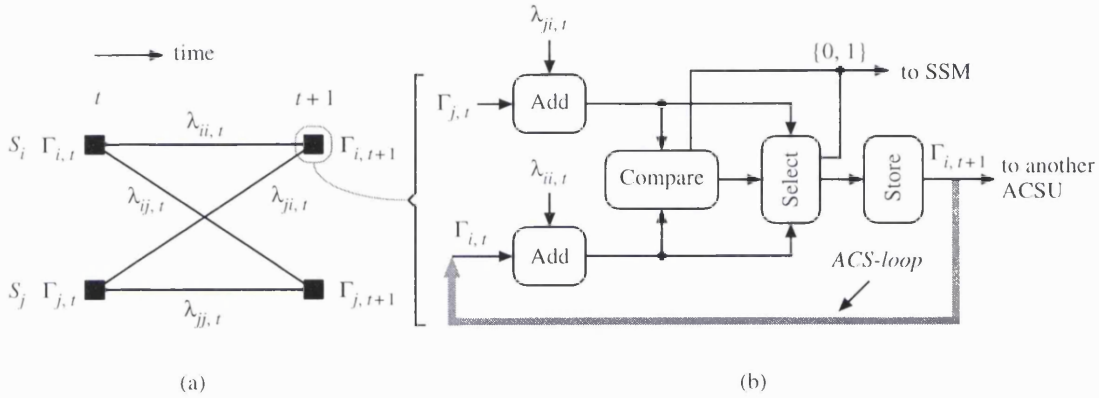


Figure 3.4 (a) Basic trellis module for a $R = 1/n$ convolutional code. (b) Block diagram of the ACSU used for the same code.

For slow Viterbi decoders a *state-serial* architecture [51] is adopted where one ACSU (or a few) is time-shared. For high-speed Viterbi decoding a *state-parallel* architecture [52] is employed where 2^{K-1} ACSU's are working in parallel. Generally, the maximum operating speed in a state-parallel implementation is limited by the propagation delay around the *ACS-loop*. Therefore, the design of the ACSU is crucial in determining the key performance parameters of a Viterbi decoder, that is, speed, complexity, size and power consumption.

As seen from Eq. (3-4) unbounded growth of the path metrics is an inherent problem with the VA. To avoid path metric overflow and to reduce the required dynamic range in internal calculations, several methods for normalising the survivor metrics have been considered [53]. The complexity involved in the normalisation process depends on which design solution is adopted. One solution requires that at each time instant the smallest survivor metric be subtracted from all others, thus forcing the minimum metric to remain at around zero. However, this solution requires extra circuitry for selecting the smallest

path metric, and hence it is not usually adopted in practice. Instead, the threshold technique is usually employed where a fixed threshold is subtracted from all path metrics when they all exceed a threshold value. The threshold technique may be realised by taking an average over all survivor metrics in each time instant and subtracting it from all path metrics when the average exceeds a threshold [13]. The threshold technique may be extended to selecting any arbitrary state and periodically monitoring its path metric. When the path metric at that state exceeds a threshold value, the amount of the difference is subtracted from all path metrics, and thus the path metric of the monitored state is forced to remain around the threshold. Finally, in digital realisations the extra computation required for metric normalisation may be easily avoided by using 2's complement arithmetic to implement Eq. (3-4) [54].

3.2.3 The SSM Block

The SSM block is responsible for keeping track of the information bits associated with the survivor paths computed by the ACS block. The two fundamental design approaches for realising the SSM block are the *register-exchange* and *traceback* techniques. In both techniques each state in the trellis diagram is assigned a shift register. The number of symbols that a shift register must be capable of storing depends on the survivor depth W, which in practice is usually chosen to be about $5K$.

In the register-exchange [20] technique, the information symbols associated with the survivor path for each state are stored directly and the registers in which these symbols are stored are interconnected in the same order as the ACSU's. Each time a new n-symbol block is processed by the decoder, the contents of the registers are updated and exchanged depending on which paths survived the ACS comparisons. A new symbol is then added at one end of each shift register, and the oldest symbol in each shift register is delivered to the OD block. The major disadvantage of this technique is that the complete history of the survivor path for each state has to be updated at each time instant. In addition, the registers must be capable of sending and receiving strings of symbols to and from one of two different locations. At high speeds all the exchanging must take place simultaneously, resulting in very complex hardware realisation. Therefore, this technique is usually only suitable for relatively slow or small (up to about $K = 5$) decoders. However, from a functional point of view this technique is very simple and does not require any trace back to find the decoded bits. That is, at time t the information symbol

associated with the branch at time $t-W$ is read directly as the oldest bit of the selected shift register. Fig. 3.5 shows the contents of the registers for the VA decoding example in Fig. 3.2 assuming register-exchange realisation.

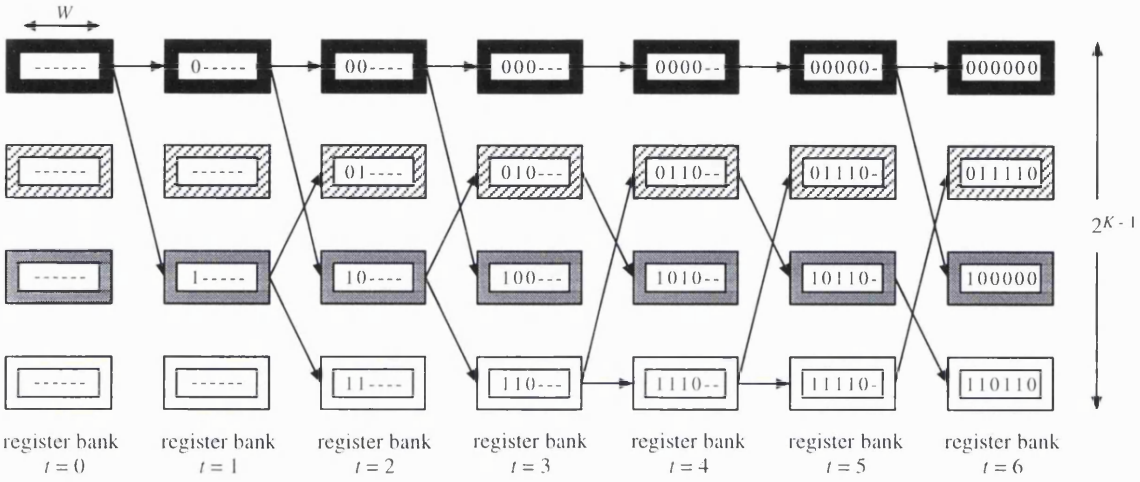
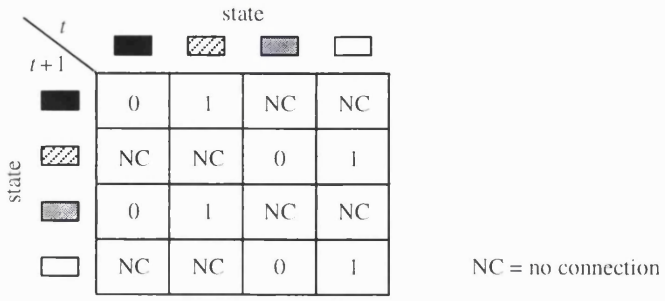


Figure 3.5 Path memory contents for the Viterbi decoding example in Fig. 3.2 using the register-exchange approach.

The traceback technique [55] is more commonly utilised and uses *pointers* to keep track of the survivor paths. That is, at each time instant there is no need to store the information bits associated with the survivor path at each state, but instead only the results of the ACS comparisons are stored. The decoder outputs information bits by selecting a shift register at time t and recalling the pointers in the reverse order in which they were stored. The information bits associated with the branch at time $t-W$ are then output. At high speeds this technique enables decoding of several branches at each trace-back. The pointers in the shift registers are updated in a first-in-first-out manner and are continuously overwritten. For the trellis diagram in Fig. 2.7, Table 3.1 represents the pointers assigned for the transitions from time t to time $t+1$. Based on this truth table, Fig. 3.6 shows the

Table 3.1 Time transition pointers for the trellis diagram in Fig. 2.10.



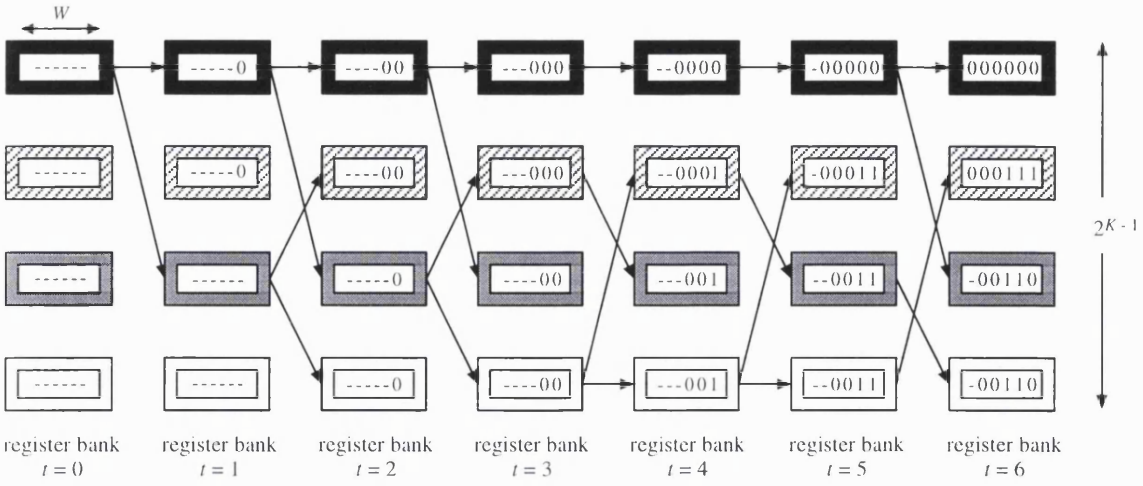


Figure 3.6 Path memory contents for the Viterbi decoding example in Fig. 3.2 using the traceback approach.

contents of the registers for the VA decoding example in Fig. 3.2 assuming traceback realisation.

3.2.4 The OD Block

The OD block determines the decoder output and it also provides information to the decoder synchroniser. The implementation complexity of the OD block depends on which design solution is adopted. At a given time t the decoder must output the information bits associated with one of the branches at time $t-W$. Because at time t there are still $2^{k(K-1)}$ survivor paths to choose from, the decoder can either select the one with the largest metric, or select one at random [20]. The first solution offers optimum coding performance and always results in the lowest probability of error but at the expense of increased complexity. In this case the OD block is essentially a WTA network[2] [19]. For the second solution, although the realisation is simple, only moderate values of CG are usually achieved. However, as W is increased, a point is reached beyond which all paths through the trellis coincide and hence the difference in performance between the two approaches becomes negligible. In practice, computer simulations are usually used to determine the optimum combination of W and decision technique.

2 The function of the WTA network is to select and identify the largest variable from a specified set of M competing variables and inhibit the remaining $(M-1)$ variables.

3.3 Applications of the VA

In its most general form, the VA may be viewed as a solution to the problem of maximum *a posteriori* probability estimation of the state sequence of a finite-state discrete-time Markov process observed in memoryless noise [2]. A finite-state Markov system has a finite number of S internal states (the state space is $Z = \{S_i,\ i = 0, 1, \ldots, S-1\}$), among which the system evolves in time. These systems are described by trellis (and state) diagrams and the VA is employed to find the most-likely noiseless finite-state sequence through a particular trellis, given a sequence of finite-state symbols that are corrupted by noise. The trellises can describe a trellis code [56], a TCM scheme [26], a continuous phase modulation (CPM) system [57], a partial-response (PR) channel [58], a language [2], and so on. In such applications, the VA is used to perform MLSD in real-time. Here, the application of the VA to digital magnetic recording systems [59], which to date has been the primary application of analogue Viterbi detection is described.

3.3.1 Applications in Digital Magnetic Recording Systems

Over the past 10 years or so, mass storage of information has been dominated by digital magnetic recording systems, mainly because these systems provide a more economical solution to high bit-capacity than semiconductor memories and optical storage [60]. A digital magnetic recording system can be regarded as a PRS system because of its inherent differentiation in the read process [59]. In outline, PRS is a multilevel baseband transmission technique for bandwidth-limited channels that deliberately generates ISI[3], and a degree of correlation between transmitted symbols (or signal levels), and so it is also referred to as *correlative coding* [61]. The introduction of ISI usually serves for the purpose of spectrum shaping which makes possible better use of the available bandwidth. This ISI is known and can be removed at the receiver. The benefits of PRS are exploited in modern magnetic recording systems to increase storage capacity.

Traditionally, the stored information from a magnetic storage medium (e.g., a computer hard disk) was retrieved using peak detection (or symbol-by-symbol) techniques [62]. Typically with peak detection a combination of a threshold and a zero-crossing detector is used to detect the read-back signal peaks. However, in order to keep the error rate below a

[3] ISI arises in pulse-modulation systems whenever the effects of one transmitted pulse are not allowed to settle entirely before the transmission of the next.

certain level, there must be sufficient separation between the peaks of adjacent symbols to avoid excessive ISI and severe performance degradation. As a result, a limit on the density of the storage system has been imposed. However, viewing the read signal as a PR signal and employing MLSD, more interference between adjacent transitions is allowed. Thus, transitions can be packed more closely on the disk, adding to its storage capacity and also increasing data rates [10]. In addition, this technology yields considerable improvements in S/N gain compared with peak detection.

A typical block diagram of a modern digital magnetic recording system employing PR-MLSD is shown in Fig. 3.7. The system consists of the write channel, the physical channel (magnetic media) and the read channel and its primary objective is to reliably store as much information as possible in a given area with typical error rates of 10^{-8} or better [62]. Types of noise encountered here include thermal noise and interference from adjacent tracks or previously recorded information, which are usually considered to be Gaussian random signals.

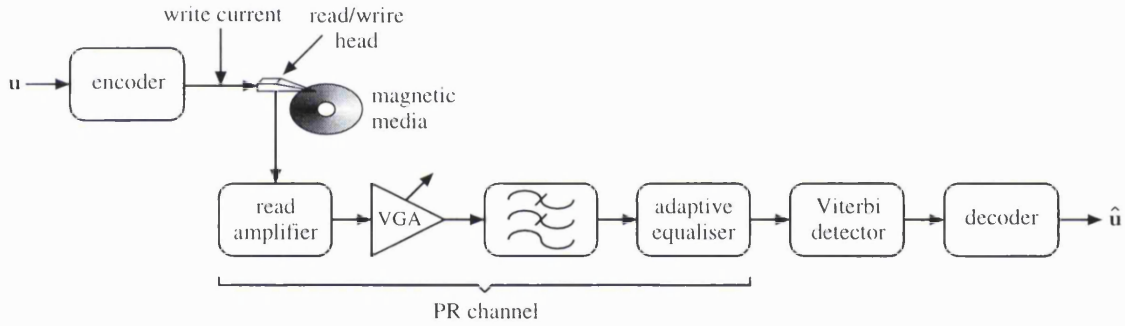


Figure 3.7 A typical digital magnetic data-storage system.

In the write process (write channel) the information sequence **u** typically undergoes coding operations such as error-correction coding for high immunity against noise and run-length-limited (RLL) coding for reliable timing extraction and reduction in ISI. The encoded sequence is then applied to the write head that is driven by a current source. Depending on the encoded sequence, the current into the write head induces a magnetisation pattern on the magnetic disk immediately below the write head. The current into the write head is configured to have only two possible values, for example $+A$ and $-A$, where the amplitude A is made sufficiently large so as to completely magnetise the magnetic media in the positive or negative direction.

During the read process (read channel), the read head senses the magnetic flux changes on the magnetic disk that occur in between the areas of positive and negative magnetisation and generates an output voltage signal which is in the μV range. This signal is first boosted by a low-noise preamplifier with a fixed-gain of about 40dB and then by a variable-gain amplifier (VGA) to a nominal level of usually 1V peak-to-peak [63]. Next, unwanted high-frequency noise is removed with a continuous-time low-pass filter followed by a finite-impulse-response (FIR) filter which shapes the spectrum of the signal to the PR target and reduces unwanted ISI. PR polynomials suitable for digital magnetic recording systems are characterised by [64]

$$F(D) = (1-D)\cdot(1+D)^{l}, \qquad l = 0, 1, 2, \ldots \tag{3-11}$$

where D is a delay operator. Out of these the three that have been proposed for disk-drive storage systems are summarised in Table 3.2. These are listed in terms of increasing recording density and the equivalent transition responses are shown in Fig. 3.8.

Table 3.2 PR polynomials for magnetic disk-drive systems.

l	$F(D)$	PR system	Normalised signal levels	States	SNR loss*
1	$(1-D)(1+D)$	PR4 (class-IV)	$(-1, 0, +1)$	4	0
2	$(1-D)(1+D)^2$	EPR4	$(-1, -1/2, 0, 1/2, +1)$	8	0
3	$(1-D)(1+D)^3$	EEPR4	$(-1, 2/3, -1/3, 0, 1/3, 2/3, +1)$	16	2.2dB

*Due to increase in ISI [64].

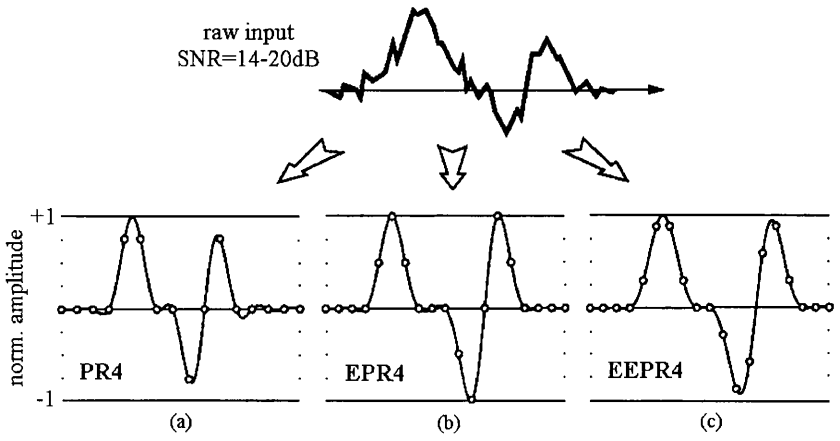


Figure 3.8 Common PR targets used in digital magnetic recording.

It should be noted that the channel with polynomial $F(D)=1-D$ is called a *dicode* channel. After equalisation the PR signal is passed on to the Viterbi detector the complexity of which depends on l since it defines the number of trellis states ($S=2^{l+1}$). The simplest is PR4 Viterbi detection, which uses interleaved complementary signals (one containing even samples and the other odd) to represent the symbol sequence[4]. Thus, instead of a 4-state Viterbi detector running at the symbol rate, there are two, 2-state Viterbi detectors each running at half the symbol rate. Such simplifications are not however directly applicable to the *extended* PR schemes. Consequently, EPR4 and EEPR4 systems require a full 8-state and 16-state Viterbi detector, respectively[5]. Finally, in order to recover the estimated sequence $\hat{\mathbf{u}}$, the output of the Viterbi detector is passed on to the decoder.

3.4 Sequential Decoding

The major drawback of the optimum VA is that since all of the possible state transitions are checked at each time instant, implementation complexity and computational effort increase approximately exponentially with increasing constraint length. Therefore, as mentioned in Section 3.1 Viterbi decoders are limited to small constraint length ($K \leq 9$) convolutional codes and are therefore, used where a moderate error probability is sufficient (about 10^{-5}).

Sequential decoding [43] approaches the performance of ML decoding by only checking some of the tree branches. The number of branches searched depends on the channel characteristics but is essentially independent of constraint length, thus making possible the use of very large constraint length (e.g., $K=41$) codes, resulting in arbitrarily low error probabilities. The main sequential decoding techniques are the *stack* and *Fano* algorithms, or variations of these.

In the stack algorithm [66], an ordered list or stack of previously examined paths of different lengths is kept in storage. Each stack entry contains a path together with its metric, and the paths are arranged in order of decreasing metric values. At each iteration,

---

[4] A PR4 system results from time-interleaving two separate dicodes; $(1-D)(1+D) \equiv (1-D)+D(1-D)$.
[5] Actually, it is possible to employ interleaving techniques to these systems but this reduces the achievable S/N gain [65].

the top path in the stack is extended by computing the branch metrics of its 2^k subsequent branches, and then adding this to the metric of the top path in the stack to form 2^k new paths. The top path in the stack is then erased, 2^k new paths are inserted, and the stack is arranged in order of decreasing metric values. All paths with metrics that fall below some preselected amount from the metric of the top path can be deleted. When the length of the top path in the stack equals the length of the received sequenced (in terms of branches), the algorithm terminates, releasing the top path in the stack as the decoded sequence. Notice that the algorithm does not necessarily advance by one branch through the tree at each iteration because of the reordering of the stack. Consequently, buffering of the received n-symbol blocks is required while the search is proceeding. Depending on the *speed factor*[6] of the decoder, long searches can cause the input buffer to overflow, resulting in some loss of data.

The Fano algorithm [67] sacrifices some speed compared to the stack algorithm, but requires essentially no storage. The Fano algorithm searches for the most-likely path through the tree by examining one path at a time, thus eliminating the storage of all but one path and its metric. Basically, the algorithm continues to move forward through the tree as long as the metric of the path being examined is growing. Whenever the metric decreases below a threshold, the algorithm backs up, and searches other paths stemming from previous nodes (states) on the already travelled path. If no path is found whose metric lies above the threshold, the threshold is relaxed and the algorithm attempts to move forward again with a lower threshold. Notice that on each successive forward search of a given node, the threshold is lower than on the previous visit to that node. This prevents looping in the algorithm and ensures that the algorithm will eventually reach to the end of the tree and terminate. Since the Fano algorithm requires no storage (apart from the input buffer), it is preferred over the stack algorithm in practical implementations of sequential decoding.

The most undesirable feature of sequential decoding is that the number of computations required is a random variable, and decoding times occasionally exceed some upper limit, causing information to be incorrectly decoded or erased [21]. In addition, unlike Viterbi decoding, the computational complexity of sequential decoding requires a significant increase for soft-decision decoding. Sequential decoders are finding application in

---

[6] Speed factor is defined as the ratio of the speed at which computations are performed to the speed of the incoming n-symbol channel blocks [21].

extremely noisy environments such as deep space and satellite links where large values of CG are required. They are also used in hybrid FEC-ARQ communication systems, where those blocks in which buffer overflows occur can be retransmitted.

3.5 Feedback Decoding

A third alternative to the optimum VA is a decoding technique called *feedback decoding* [44], which was initially developed for hard-decision decoding. In feedback decoding, the decoder makes a decision on a given k-bit information block based on metrics computed over *only one* constraint length (or slightly more) of the received sequence (i.e., $W = K$). Since each k-bit information block influences no more than a block of Kn code symbols (see Section 2.3.1), from a feedback decoder's point of view, decoding this information block requires only the examination of the Kn symbol span of the received sequence that is influenced by it. It is possible, therefore, to view the decoder as *a sliding block decoder*, each Kn-symbol block of the received sequence corresponding to a k-bit information block, and offset by n symbols from its neighbours.

In outline, the operation of a feedback decoder is as follows. Starting from a given state, the decoder observes all paths to a depth K into the tree, and compares all 2^K paths[7] of this *sub-tree window* with the corresponding sliding block or the received sequence. The decoder then selects the path with the largest (or smallest) metric and outputs the information bit associated with the first branch of this path. In fact, there is no need to trace back the "winning" path (as required in the VA) for the decoded bit, but instead only to establish whether the path originated in the upper or the lower half of the sub-tree window examined. If the selected path lies in the upper half of the sub-tree window, the decoded bit is a "0". Otherwise the decoded bit is a "1". The current decoding decision and the previous starting node define the new starting node for the next decoding decision. This is equivalent to "feeding back" the decisions to the decoder in determining the sub-tree window to be considered next. Hence, upon receiving the next n-symbol block of the received sequence, the decoder looks to all paths to depth K extending from the new starting node and makes comparisons with the next sliding block of the received sequence. The decoder may proceed in this manner indefinitely examining one sliding

[7] For simplicity but without loss of generality the code rate is assumed to be $R = 1/n$.

block at a time and decoding the information associated with it. Figure 3.9 illustrates the equivalent feedback decoding process for the Viterbi decoding example in Fig. 3.2.

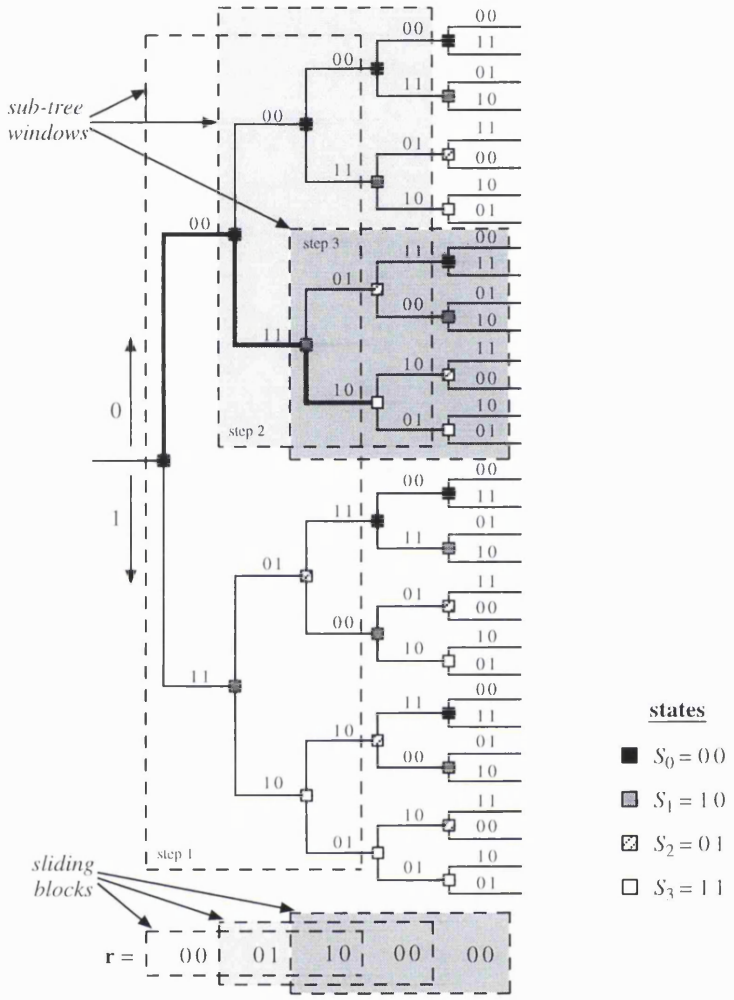


Figure 3.9 Equivalent feedback decoding processes for the Viterbi decoding example in Fig. 3.2.

3.5.1 Syndrome-Feedback Decoding

The fact that a feedback decoder can be considered as a sliding block decoder can be exploited to simplify its implementation for hard-decisions. Therefore, instead of computing metrics as described above, a feedback decoder has been implemented by computing the *syndrome sequence* from the received sequence and using a table-lookup technique to correct the channel error symbols [20]. Just as with block codes, the syndrome sequence provides linear equations that can be solved for the most-likely channel error sequence. Correction symbols for the received information symbols are formed through appropriate mapping of a span of syndrome symbols (e.g., through a

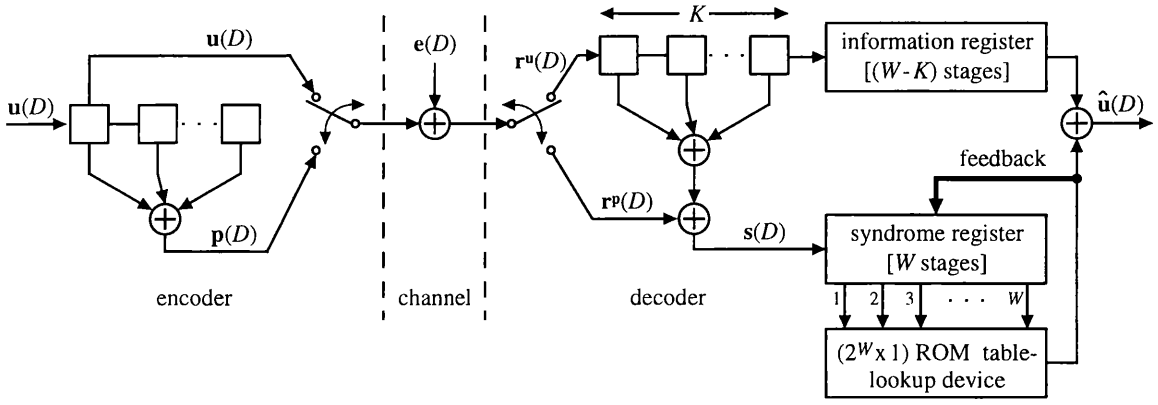


Figure 3.10 Syndrome feedback decoding system using a $R = 1/2$ systematic convolutional code.

look-up table). Figure 3.10 shows a *syndrome-feedback* decoding system employing a $R = 1/2$ systematic code. The input information sequence (polynomial representation is used) is encoded into the information sequence $\mathbf{u}(D)$ and the parity sequence $\mathbf{p}(D)$ that are multiplexed for serial transmission over the channel. If the channel error sequence $\mathbf{e}(D)$ is modelled as the information error sequence $\mathbf{e}^{\mathbf{u}}(D)$ and the parity error sequence $\mathbf{e}^{\mathbf{p}}(D)$, then the received information and parity sequences are

$$\mathbf{r}^{\mathbf{u}}(D) = \mathbf{u}(D) \oplus \mathbf{e}^{\mathbf{u}}(D) \tag{3-12}$$

and

$$\mathbf{r}^{\mathbf{p}}(D) = \mathbf{p}(D) \oplus \mathbf{e}^{\mathbf{p}}(D), \tag{3-13}$$

respectively. The received sequences are then compared by modulo-2 addition to compute the syndrome sequence $\mathbf{s}(D)$, that is

$$\begin{aligned} \mathbf{s}(D) &= \mathbf{r}^{\mathbf{u}}(D) \cdot \mathbf{g}^{(2)}(D) \oplus \mathbf{r}^{\mathbf{p}}(D) \\ &= [\mathbf{u}(D) \oplus \mathbf{e}^{\mathbf{u}}(D)] \cdot \mathbf{g}^{(2)}(D) \oplus \mathbf{u}(D) \cdot \mathbf{g}^{(2)}(D) \oplus \mathbf{e}^{\mathbf{p}}(D) \\ &= \mathbf{e}^{\mathbf{u}}(D) \cdot \mathbf{g}^{(2)}(D) \oplus \mathbf{e}^{\mathbf{p}}(D) \end{aligned} \tag{3-14}$$

which is routed to the syndrome register. From Eq. (3-14) it is observed that the syndrome is a function of the channel noise *only* and does not depend on the information sequence. Thus, in the absence of transmission errors $\mathbf{e}^{\mathbf{u}}(D) = \mathbf{e}^{\mathbf{p}}(D) = \mathbf{s}(D) = 0$ and the contents of the syndrome register are all "0"s. In the presence of transmission errors "1"s appear in the syndrome register. The contents of the syndrome register are used to address the decision device (e.g., a read-only memory (ROM) device) whose output attempts to

correct the error in the right-most symbol held in the information register. The resulting decoder output is

$$\begin{aligned}\hat{\mathbf{u}}(D) &= \mathbf{r}^{\mathbf{u}}(D) \oplus \hat{\mathbf{e}}^{\mathbf{u}}(D) \\ &= \mathbf{u}(D) \oplus [\mathbf{e}^{\mathbf{u}}(D) \oplus \hat{\mathbf{e}}^{\mathbf{u}}(D)]\end{aligned} \tag{3-15}$$

where $\hat{\mathbf{e}}^{\mathbf{u}}(D)$ is the estimate of $\mathbf{e}^{\mathbf{u}}(D)$. In outline, each time a new symbol enters the syndrome register, its contents are compared with the 2^W possible error patterns stored in the ROM, and the most-likely one is selected. If the selected error pattern contains an error (i.e., a "1") in the symbol corresponding to the right-most position of the information register, then the decision device outputs a "1" thereby correcting the error. Otherwise the device outputs a "0". The correction symbol is also fed back to the syndrome register to remove its effect on the span of syndrome symbols to be examined next by the decoder.

3.5.2 Error Propagation Effects

As long as the estimated error symbols are correct, the syndrome-feedback decoder (or any other feedback decoder) continues to operate properly correcting errors in the received sequence. However, when an incorrect estimate is made, not only an incorrect decoded symbol is read out, but also the incorrect estimate causes subsequent decoded symbols to be in error. This is called the *error propagation* effect[8] of feedback decoders and occurs even when the code itself is noncatastrosphic [68].

The worst situation appears when the error propagation is *unlimited* in which case catastrophic failure is unavoidable. Fortunately, various solutions have been suggested to prevent this phenomenon. An obvious solution to limiting error propagation is to resynchronise the decoder periodically by inserting a string of $k(K-1)$ "0"s into the information sequence after every kL information symbols. This periodic resynchronisation limits error propagation to at most $(L+K-1)$ time instants. However, for the fractional rate loss $(K-1)/(L+K-1)$ to be small, L must be much larger than K (see Section 2.3.1), and so the error propagation span can still be quite long. The effects of error propagation can also be limited by using a convolutional code that has automatic

[8] Error propagation is to some extent common to all convolutional decoding algorithms [27].

recovery properties [69]. For such codes the decoder recovers from error propagation upon receipt of a relatively short span of error-free channel symbols.

A better approach to limiting error propagation is to make $W > K$. The effect of W upon error propagation when a feedback decoder is employed with the code in Fig. 2.7, is illustrated in Fig. 3.11. The diagrams in this figure show the state transition probabilities for $W = 2, 3, 4, 5 \& 6$ assuming the all "0"s sequence is transmitted. It is clear from the diagrams that for $W = 2$ or $W = 3$, it is impossible to reach state S_0 once an error has occurred. This leads to unbounded error propagation. For $W = 4$ there is a 0.5 probability that once the decoder is at state S_2 it will reach state S_0. In this case the error propagation span after channel errors have ceased is also unbounded. For $W = 5$ the error propagation span is unbounded too. However, for $W = 6$ all paths lead to state S_0 within two branches and thus the error propagation span is bounded. It has been shown [68] that when using a minimum distance feedback decoder with any $R = 1/2$ noncatastrophic convolutional code, the necessary and sufficient condition to prevent unlimited error propagation is

$$W \geq K\left(\left\lfloor \frac{K}{2} \right\rfloor + 1\right) \tag{3-16}$$

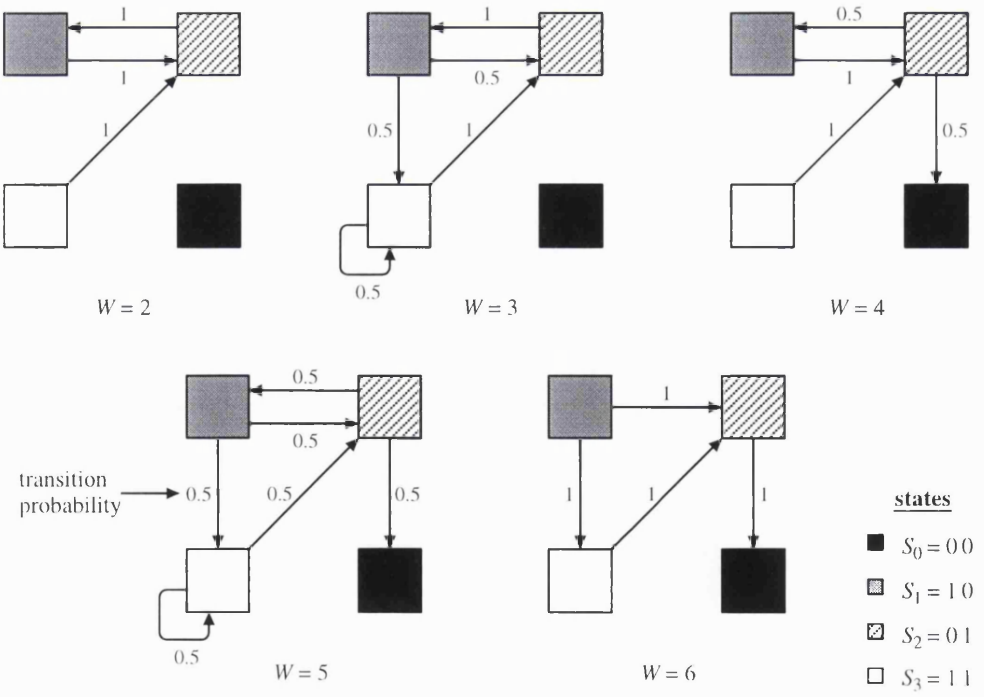


Figure 3.11 Effect of W upon error propagation when feedback decoding the code in Fig. 2.7.

where $\lfloor x \rfloor$ denotes the largest integer contained in x.

Error propagation can be completely prevented by simply breaking the decision feedback connection in a syndrome-feedback decoder. This type of decoding is called *definite decoding* [70]. However, the elimination of the decision feedback loop usually decreases the error-correcting capabilities of the code. Thus, generally their coding performance is inferior compared to syndrome-feedback decoding systems with the associated error propagation.

3.5.3 Feedback Decoding Performance and Threshold Decoding

Recall that in a syndrome-feedback decoder the decoding depth W is usually made equal to *only one* constraint length, hence its maximum error-correcting capability (i.e., in the absence of error propagation) is based on the $d_{\min}$ of the code. Based on this, it can be shown that on the BSC and for large E_b/N_o, the bit error performance of feedback decoding is approximated by [21]

$$P_e \approx C_v \exp\left(\frac{-Rd_{\min}E_b}{2N_o}\right) \tag{3-17}$$

where C_v is a constant associated with code structure. Since nonsystematic convolutional codes cannot achieve larger $d_{\min}$ values than systematic convolutional codes [27], no increase in error-correcting capability can be achieved by employing nonsystematic convolutional codes with syndrome-feedback decoders.

Generally speaking, the complexity of a syndrome-feedback decoder depends almost entirely on the complexity of the decision device. For reasonable values of CG a long constraint length code must be employed *or* W must be made quite large. As K (or W) gets larger, however, the size of the decision device becomes impractical, in which case Viterbi decoding would be employed. The main advantage of syndrome-feedback decoders is that interleaving and deinterleaving can easily be included as part of the encoder and decoder [44]. Thus, these systems are well suited for burst noise channels which require moderate values of CG at low-cost. Examples of such burst environments include HF radio and some types of telephone channels.

Finally, in some cases, the table-lookup device can be replaced with a threshold device, and the decoding is called *threshold decoding* (or *majority-logic* decoding) [37]. The relatively simple decision device employed in threshold decoding allows the use of large K. Unfortunately, threshold decoders achieve very small values of CG mainly due to the algebraic structures of the convolutional codes that must be employed. Soft-decision threshold decoders have also been developed [71], but the approximately 2dB improvement in CG cannot really justify the substantial increase in implementation complexity.

3.6 The Modified Feedback Algorithm

In the previous section it was mentioned that the maximum error-correcting capability of a feedback decoder is based on the $d_{\min}$ over just one constraint length of the received sequence and systematic convolutional codes are usually employed. On the other hand, a truncated Viterbi (or sequential decoder) processes at least four or five constraint lengths and thus, it bases its decisions on the d_{free} between sequences. In addition, for Viterbi decoding, nonsystematic convolutional codes are preferred. Comparing Eq. (3-5) with Eq. (3-17) and given that for a particular R and K set the maximum d_{free} of the best nonsystematic codes is about twice the maximum $d_{\min}$ of the best systematic codes [21], Viterbi decoding has roughly a 3dB advantage in terms of asymptotic CG over feedback decoding. Thus, the constraint length for feedback decoding must be about twice that for Viterbi decoding to achieve comparable performance. Alternatively, by making W much greater than K (e.g., $W \approx 5K$ as in a truncated Viterbi decoder) and employing nonsystematic convolutional codes the coding performance of feedback decoders can also be increased. Unfortunately, since implementation complexity in conventional feedback decoders increases exponentially with increasing W (or K), there appears to be no real advantage in using feedback decoding with large W, since a Viterbi decoder can provide a more efficient trade-off between performance and complexity for the same R, K and W.

A high-speed architecture for small feedback decoders based on the neural *Hamming classifier* [72] has been reported in [73] which formed the basis for the development of the MFDA presented in this section. Although the architecture in [73] theoretically offers the potential for very high speed decoding, it suffers from the fact that implementation complexity like that for a feedback decoder employing a table look-up device, grows

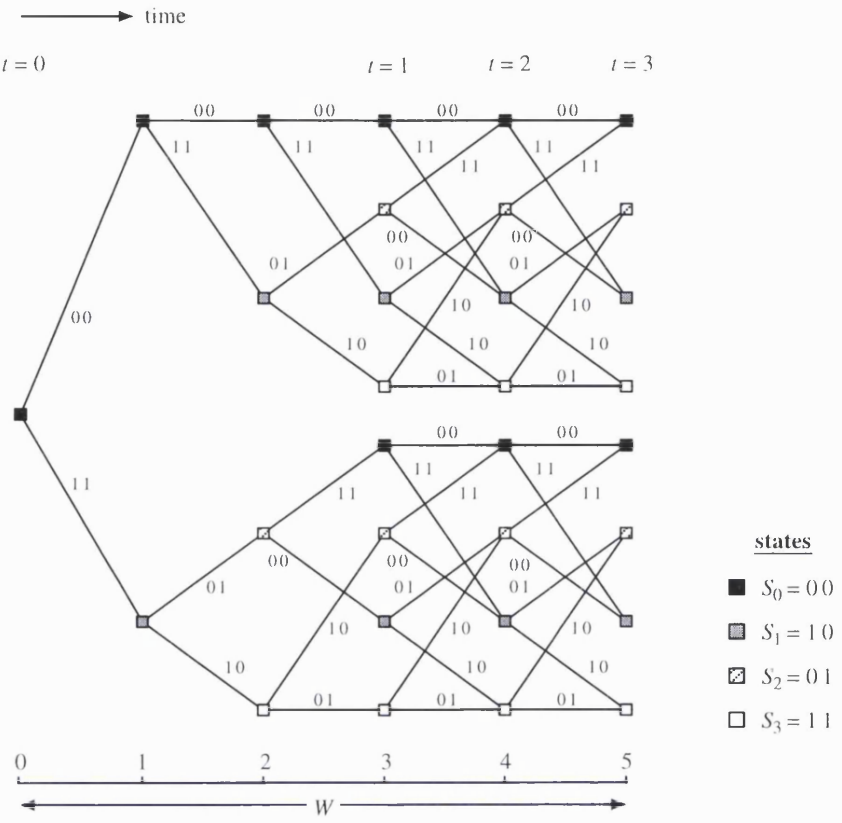


Figure 3.12 The modified tree principle.

exponentially with increasing W. Even for $W = 2K$ the amount of memory required is impractical for all but the smallest feedback decoders (i.e., $K \leq 4$).

A possible solution to the problem of memory requirement for implementing feedback decoders with $K > 4$ and $W > 2K$ (thus resulting in larger values of CG) is to employ the path elimination techniques of the VA. This is made possible by viewing feedback decoding using the tree method depicted in Fig. 3.9. The incorporation of the ACS operations of the VA into feedback decoding is accomplished by converting each sub-tree window corresponding to a particular sliding block of the received sequence into a double trellis, called the *modified tree*, as shown in Fig. 3.12 for the tree diagram in Fig. 2.9. In order to decode the information bit associated with a particular sliding block of the received sequence, the first K levels of the corresponding modified tree are processed in parallel (i.e., with no path elimination), and the remaining $(W - K)$ levels are processed serially by computing equation Eq. (3-4) in both trellises separately. In this way the number of paths is limited to 2^K (irrespective of W) and the amount of memory required is only $n(K+2)2^K$ branch symbols compared to 2^{nW} that would be required in [73].

Once all levels of the modified tree are processed, the position of the path with the largest metric among the 2^K survivors is determined, which in turn determines the decoded bit. The decoded bit is a "0" if the selected path lies in the upper trellis of the modified tree whereas is a "1" otherwise. The decoded bit is also fed back to update the branch symbols on the initial $(K-1)$ levels of the modified tree to form the modified tree corresponding to the subsequent sliding block of the received sequence. Decoding may continue in this manner indefinitely.

This convolutional decoding technique which provides true ML decoding given the constraint of finite block decoding is called the MFDA [74]. The novelty behind the MFDA is that it eliminates the need for (digital) path memory, which is not the case with the VA. By doubling the trellis, the origin of the largest path within each modified tree window can be directly established without the need for tracing back any path memory block. Unlike the architecture in [73], the implementation complexity of the MFDA is independent of W, and therefore larger values of CG are possible. The decoder corresponding to the MFDA is called the *modified feedback decoder* (MFD). Generalisation of the proposed approach to $R=k/n$ codes is possible but this is somewhat more complicated than $R=1/n$ since in practice some sort of trace back or puncturing would be required.

3.6.1 Simulated Performance

The bit error performance of the MFDA on the BSC with BPSK modulation was investigated by computer simulation (using custom written programs in C). In particular, the error-correcting capability of the MFDA was tested for different values of W for the following two optimum convolutional codes [21]:

1. $(2, 1, 3)$ with generators $\mathbf{g}^{(1)} = [1\ 0\ 1]$, $\mathbf{g}^{(2)} = [1\ 1\ 1]$, and
2. $(2, 1, 5)$ with generators $\mathbf{g}^{(1)} = [1\ 0\ 1\ 1\ 1]$, $\mathbf{g}^{(2)} = [1\ 1\ 0\ 0\ 1]$.

Figure 3.13a shows averaged simulation results (the simulations were repeated over five times) of the relation between the channel error rate p, and the bit error probability P_b for code 1, while Fig. 3.13b shows the results of similar simulations for code 2. From the curves in these figures it is observed that P_b decreases with increasing W. In other words, the error-correcting capability of the MFDA improves as the decoding depth W is

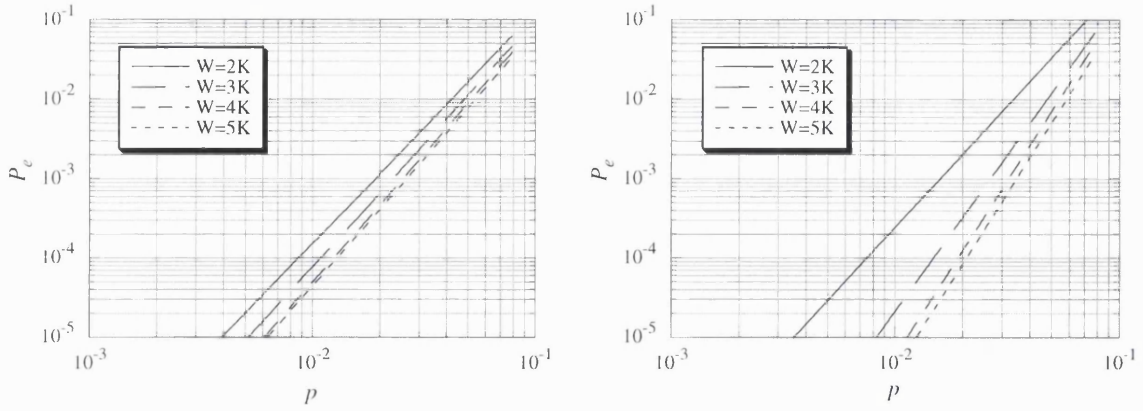


Figure 3.13 Bit error probability versus channel error rate as a function of W for a convolutional coding system employing the MFDA; (a) code 1, (b) code 2.

increased. The curves in Fig. 3.14 establish the level of CG achieved over uncoded BPSK modulation for the same two codes and for various values of W. From these curves it is clear that the CG of the MFDA also improves as W gets larger. It is expected that with soft-decision decoding an additional 2dB increase in CG should be achieved.

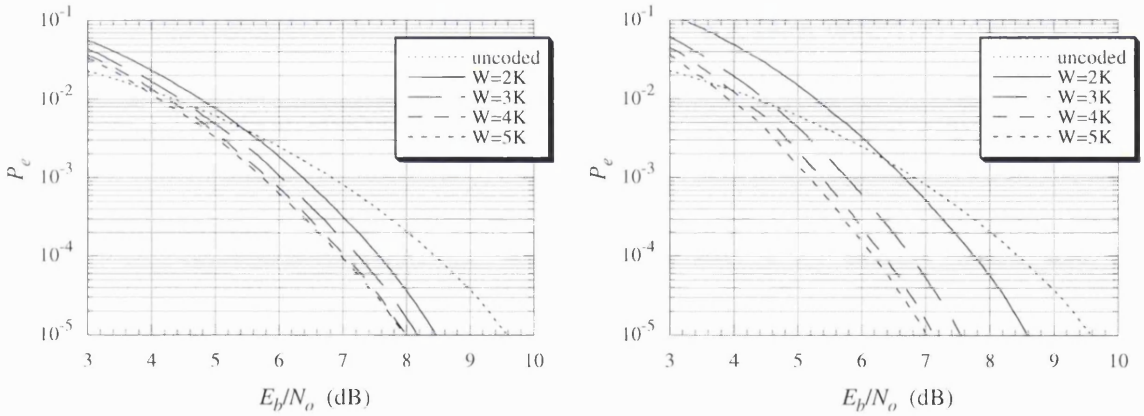


Figure 3.14 Performance of a convolutional coding system employing the MFDA as a function of W on the BSC with BPSK modulation; (a) code 1, (b) code 2.

3.6.2 Comparison with Viterbi Decoding

The MFDA embodies features of feedback and Viterbi decoding. In particular, the MFDA employs the path elimination techniques of the VA and the "decision-feedback" feature of feedback decoding. The two algorithms are compared here on the basis of their performance, decoding speed, and implementation complexity.

3.6.2.1 Performance: The MFDA can be regarded as a variant of the VA since it employs the VA principle (i.e., the ACS operations encountered in the VA). More specifically, the MFDA can be viewed as equivalent to a truncated metric and path memory realisation of the VA. Although conventional realisations of the VA truncated the path memory to a depth of about $5K$ the metric memory is not usually truncated. However, it has been shown [12] that the effects of finite metric memory on the performance of a Viterbi decoder are negligible with metric memory truncation at a depth of about $5K$. This idea is also exploited in the sliding block Viterbi decoder (SBVD) approach reported recently in [75]. In the SBVD scheme, the received sequence is decomposed into overlapping sliding blocks of depth $10K$, and for each sliding block, the VA is used to find the optimum path through a given trellis of the same length. In particular, the first $5K$ trellis levels in the SBVD scheme are used for block synchronisation because the starting state for each trellis is unknown[9], and the remaining levels form the decoding depth. Thus, upon reaching the final level of a finite length trellis, the best survivor is traced back for $5K$ to decode the information bits corresponding to that level and so path memory is required. The performance of the SBVD asymptotically approaches that of truncated Viterbi decoding.

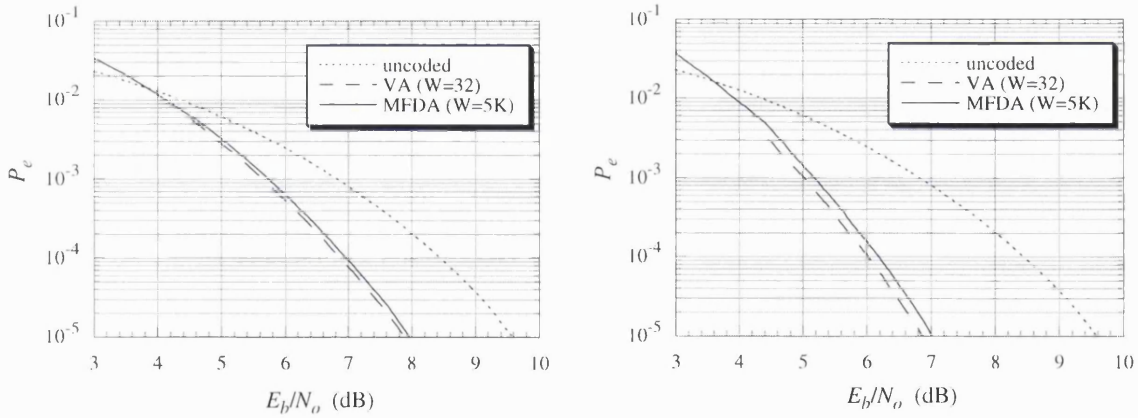


Figure 3.15 Comparison of the performance of the MFDA and the truncated VA on the BSC with BPSK modulation; (a) code 1, (b) code 2.

In Fig. 3.15 the performance of the MFDA ($W = 5K$) and the truncated path memory VA ($W = 32$) obtained from [48] are compared for the two codes considered (the decoding

[9] Note that with the MFDA the starting state of each modified tree is known.

depths are not the same). From these curves it is observed that the loss in CG when the MFDA is employed is negligible compared to the truncated Viterbi decoder. In particular, at $P_b = 10^{-5}$ the CG loss for the code 1 is only 0.075dB and for code 2 is about 0.15dB. Thus, the performance of the MFDA asymptotically approaches that of truncated Viterbi decoding.

3.6.2.2 Decoding speed: By using a state-parallel implementation a Viterbi decoder can theoretically decode information bits at the rate at which an ACS operation can be performed. This makes Viterbi decoding very attractive for high-speed applications such as satellite communications. On the other hand, in order for a MFD to make a decoding decision (again assuming a state-parallel implementation), $(W - K)$ sequential steps are required. This decoding delay encountered with the MFDA results in a loss in decoding speed compared to Viterbi decoding. One solution to improving the decoding speed of the MFDA is to increase the parallel and reduce the sequential steps involved in each decoding cycle. This trades hardware complexity against decoding speed because the number of nodes in the modified tree doubles for each unit decrease in the number of sequential steps. A less hardware intensive solution is to use the N-step trellis technique [46] shown in Fig. 3.16 that combines N transitions of the original trellis. When this technique is adopted in the MFDA the number of sequential steps would be computed N times faster. In turn, this would result in an increase in the decoding speed by a factor of N. Note that with this technique the number of nodes in the modified tree remains constant irrespective of the decrease in the number of sequential steps. However, the number of transitions entering and leaving each node increases exponentially (2^N) as N increases linearly, and hence in practice a value of $N \leq 4$ should probably be used.

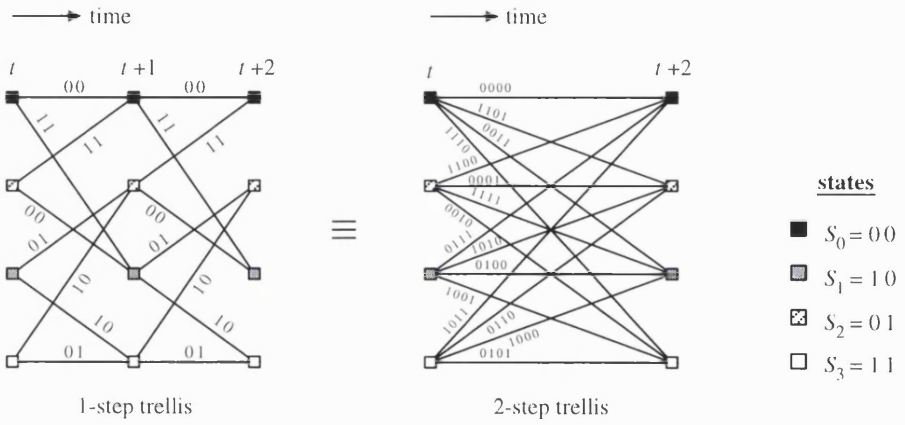


Figure 3.16 The N-step trellis principle ($N = 2$).

3.6.2.3 Implementation complexity: As mentioned in Section 3.2 a practical Viterbi decoder requires two types of memory. One is the metric memory that stores the metrics of the survivors and the other is the digital path memory (SSM block) that stores the history of the survivor paths in order to enable trace back of the most-likely path at each time instant. Although the realisation of the metric memory is fairly simple (either an analogue or a digital implementation can be adopted), the realisation of the path memory is quite complicated mainly because of the operations that it executes. Generally speaking, the SSM block accounts for a large part of the circuit complexity of a Viterbi decoder and hence its realisation is expensive in terms of die area (typically about 50% of the total [76]) and power consumption. On the other hand, for the implementation of the MFDA only a metric memory is required since there is no trace back in the decoding process. As a result, the MFDA lends itself especially well to an analogue implementation for small size and low-power dissipation. Such a decoder would be particularly desirable in applications where the data rate and power requirements are incompatible with the use of a DSP and an ASIC is required. Thus, the MFDA offers a better trade-off between speed, complexity and power dissipation for low to moderate data rate (up to about 5Mbit/s) applications of the type described in Section 2.4.

3.7 Considerations for the Implementation of the MFDA

The general block diagram of a MFD is shown in Fig. 3.17. It is composed of the symbol storage (SS) block, the BMC block, the ACS block, and the WTA network. The decoder is also provided with a symbol-timing clock so that it can distinguish between separate symbols. Only the blocks that are significantly different from those employed in a Viterbi decoder are discussed.

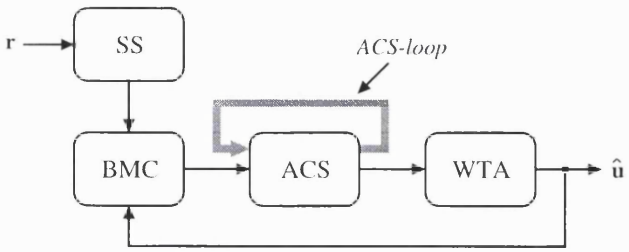


Figure 3.17 Block diagram of a MFD.

3.7.1 The SS Block

The function of the SS block is to store the received sequence **r** for a window of depth W branches (i.e., one sliding block). It typically consists of nW symbol storage elements, which can be either a shift register for hard-decision decoding or S/H circuits (i.e., an analogue delay line) for soft-decision decoding. For each decoding cycle, the last nK symbols in the SS block are used for the parallel computations while the remaining $n(W-K)$ symbols are partitioned into n-symbol blocks that are selected one at a time by a parallel-to-serial multiplexer for the sequential computations.

3.7.2 The BMC Block

The BMC block correlates the stored symbols in the SS block with the branch symbols on the associated W-level modified tree diagram to produce the corresponding branch metrics. Squared Euclidean distance or Hamming distance as in the VA (see Section 3.2.1) are employed. Unlike the VA, not all branch symbols in the MFDA are fixed but some require updating between applications of each sliding block of the received sequence. This is because the origin of the modified tree (or sub-tree in the feedback decoding case) is altered according to the previous decoding decision. As discussed in Section 3.6, only the branch symbols in the first $(K-1)$ levels of the modified tree need to be updated between applications of sliding blocks.

The dependence of these symbols on the previous decoding decision relies on the specific convolutional code employed and can easily be derived by inspection. The general form

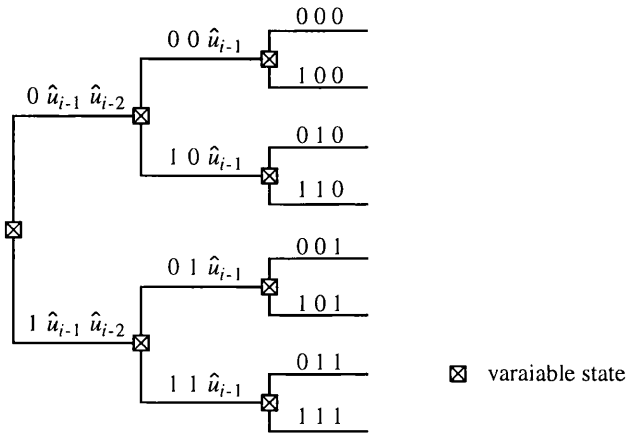


Figure 3.18 General form of the sub-tree window for the code in Fig. 2.7.

of the sub-tree (over the first 3 levels) for the convolutional code in Fig. 2.7 is shown in Fig. 3.18 where $\hat{u}_{i-1}$ represents the previous decoded bit and $\hat{u}_{i-2}$ the one before that. Note that the symbols that appear on the branches of this diagram are the contents of the encoder shift register because this makes it easier to derive the dependence talked about. The combinatorial transformation of these symbols into actual branch symbols is specified by the generator sequences of the code (see Section 2.3.1).

3.7.3 The WTA Network

The WTA network determines the decoder output, which in turn updates some branch symbol entries in the of BMC block in the manner described in Section 3.7.2. Once all W levels of the modified tree corresponding to a particular sliding block of the received sequence are processed, the WTA is instructed to determine whether the path with the largest metric (among 2^K survivors) lies in upper or lower half of the modified tree. The position of this path in turn directly establishes the decoder output. The decoded bit is a "0" if the path lies in the upper half of the modified tree and a "1" otherwise. This terminates the decoding cycle after which all survivor path metrics are reset to zero and the decoded bit is fed back to the BMC block.

Chapter

4

Implementation Using Current-Mode Analogue Circuit Techniques

IN this chapter the current-mode approach is investigated as an alternative to its voltage-mode counterpart for implementing certain sections of general classes of Viterbi decoder. The chapter begins with a brief review of the various approaches used for the realisation of Viterbi decoders, and emphasises the advantages of the *hybrid analogue/digital* approach over conventional digital techniques. Following this, a CMOS current-mode circuit for realising the fundamental building block of a Viterbi decoder is presented together with circuit enhancements in BiCMOS technology. Related issues such as maximisation of dynamic range, decoder front-end and distance computation circuitry are also dealt with. The current-mode approach is also extended to the realisation of the MFDA where suitable circuits are presented.

4.1 Realisation of Viterbi Decoders

In practice, the approach employed for the implementation of the VA depends primarily on the data rate requirement of the particular application. For low data rates (up to about 100kbit/s) a DSP based Viterbi decoder realisation provides the most economical solution whereas for medium to high data rates an ASIC Viterbi decoder is required. Generally,

the choice between a DSP and an ASIC depends on different constraints as listed in Table 4.1.

Table 4.1 A comparison between a DSP and an ASIC.

Parameter	DSP	ASIC
Flexibility/programmability	Very high	Low
Processing power	High	Very high
Speed	High	Very high
Design time	Short	Long
Design cost	Low	High
Power dissipation	High	Low

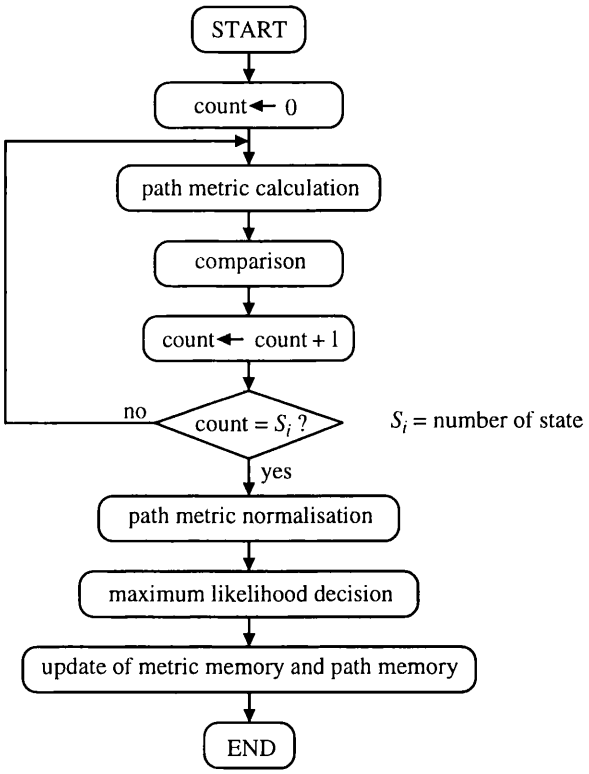


Figure 4.1 Viterbi decoding flow diagram on a DSP.

4.4.1 DSP Realisation

The realisation of a Viterbi decoder on a DSP is based on a state-serial architecture. Nowadays, the architectures of many modern DSP's such as the TMS320C54x family of DSP's from Texas Instruments, Inc. [77] are optimised for Viterbi decoding. This provides an economical solution to realising the VA for low data rate applications of the type described in Section 2.4. The maximum achievable data rate of a DSP Viterbi decoder depends not only on the architecture and clock cycle (MIPS) of the DSP but also

on the number of encoder states. A typical decoding speed for a 64-state DSP Viterbi decoder is about 10kbit/s [76]. The Viterbi decoding flow diagram for DSP implementations is shown in Fig. 4.1.

4.4.2 Digital Realisation

For medium to high data rates (i.e., 1-200Mbit/s) some parallelism must be introduced into the architecture of a Viterbi decoder, and the circuit realisation is usually based on digital techniques [52], [78]-[81]. The architecture of a medium speed Viterbi decoder is based on *area-efficient* techniques [82], which trade speed for silicon area. On the other hand, a high-speed Viterbi decoder is based on a state-parallel architecture and in a digital realisation the number of gates in the ACS and SSM blocks dominate the total number of gates in the decoder. For example, the number of gates versus the number of states for a high-speed $R = 1/2$ digital Viterbi decoder realised in 1μm CMOS process is shown in Fig. 4.2 [76]. Given that for $R = 1/2$ and $K = 7$, the ACS block and the SSM block share approximately 50% of the total Viterbi decoder hardware, then from Fig. 4.2 is observed that each digital ACSU requires approximately 400 gates. High-speed CMOS digital Viterbi decoders with $K = 7$ are commercially available and Table 4.2 summarises their key performance parameters.

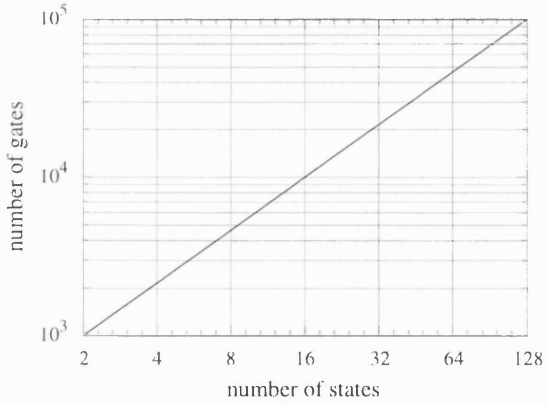


Figure 4.2 Digital Viterbi decoder hardware size for $R = 1/2$ (CMOS 1μm). Reproduced from [76].

To achieve even higher data rates (i.e., break the ACS-loop speed bottleneck) increased parallelism must be introduced and data rates up to 1.2Gbit/s were recently reported [75], [83]. However, the power dissipation of these decoders is prohibitively high.

Table 4.2 Some commercially available digital high-speed CMOS Viterbi decoders.

Company	Part name	R	R_p	Data rate* (Mbit/s)	Power Dissipation (W)†
Qualcomm, Inc.	Q1900	1/2 , 1/3	3/4 , 7/8	25	0.8
Stanford Telecomm, Inc	STEL-2060	1/2	2/3 - 7/8	25	Not known
Comatlas	CAS 5083	1/2	2/3 - 8/9	30	Not known
Motorola, Inc	MC92300	1/2	2/3 - 7/8	25	Not known

*For $R = 1/2$ (higher data rates are achieved for R_p), †maximum.

4.1.3 Analogue Realisation

In an attempt to reduce the power dissipation and increase the speed of Viterbi decoders, the realisation of certain sections of these systems (i.e., the BMC, ACS and OD blocks) using analogue circuit techniques has been proposed [84]-[86], resulting in a *hybrid analogue/digital* system. In an analogue Viterbi decoder (as commonly referred to in the literature), samples of the received analogue signal are stored by sample-and-hold (S/H) circuits, and all the arithmetic operations in the BMC and ACS blocks are performed in the analogue domain. The SSM block is realised using digital components as in a digital Viterbi decoder.

Recently, analogue Viterbi decoders have become particularly popular in PR disk drive applications employing PR4 equalisation [5]-[8]. In this application, analogue Viterbi decoders have proven to offer smaller, faster (operation in excess of 200Mbit/s), and less power hungry designs than their digital counterparts [4]. This is due mainly to the elimination of the 6-bit flash A/D converter [87] at the front-end of the decoder, which generally is a complex and power hungry component. Such decoders are currently found in today's state-of the-art commercial realisations of magnetic read channels [10]. However, PR4 detection allows simplifications during Viterbi detection, which can not be applied to general Viterbi detection. In particular, PR4 analogue Viterbi decoders employ time-interleaving which naturally doubles the operating speed and simplifies implementation complexity. Thus, instead of one four-state Viterbi detector running at the symbol rate, there are two two-state Viterbi detectors running at half the symbol rate. In addition, PR4 Viterbi detectors employ the so-called *difference-metric algorithm* [4], which allows further reduction in the implementation complexity since metric normalisation is completely avoided. Finally, another advantage of PR4 Viterbi detectors is that when an analogue realisation is employed the ACS-loop is no longer required since

only combinations of the current and previous sampled inputs with a DC offset are required [7], [8]. Thus the time delay imposed by the ACS-loop in general classes of Viterbi decoder is not an issue in analogue PR4 Viterbi detectors.

Nevertheless, the benefits gained through PR4 analogue Viterbi decoder realisations (i.e., lower power dissipation, smaller size and higher operating speed than digital PR4 Viterbi designs) can still be maintained in general classes of Viterbi decoder (e.g., EPR4) as pointed out in [13]. This is possible because of the following two reasons:

- The resolution required by the signal processing techniques in the ACS and BMC blocks of a Viterbi decoder is usually in the order of 3 to 6 bits and the computations required are simple. Since high accuracy is not required, simple analogue circuits can be employed to perform these mathematical operations.

- Analogue Viterbi decoders require no A/D converter front-end interface, which is the case for digital realisations. Thus, analogue Viterbi decoders lend themselves especially well to soft-decision decoding.

Implementing general classes of Viterbi decoder in the analogue domain was first suggested and investigated in [12]. The main objective there was to demonstrate the potential for high speed Viterbi decoding (at data rates in excess of 50Mbit/s) using analogue discrete components. In particular, voltage-mode analogue circuit techniques were employed for the implementation of the BMC and ACS sub-sections of a Viterbi decoder while ECL (emitter-couple logic) components were employed for the implementation of the SSM block. However, the circuit techniques employed in [12] are not suitable for integration as was pointed out in [5].

More recently other attempts to implement general classes of Viterbi decoder in the analogue domain have been reported [13], [88]. The technique proposed in [88] is essentially limited to hard-decision decoding since extension to soft-decision decoding results in a large increase in implementation complexity and requires high-voltage supply rails. In addition, it does not eliminate the need for an A/D converter, which is a primary motivation for an analogue realisation [84]. The approach presented in [13] although highly suitable for realising general classes of Viterbi decoder, is more oriented towards the disk-drive industry where in most cases the crucial parameter is maximum operating speed rather than low-power dissipation. This approach employs BiCMOS voltage-mode

circuit techniques, has a power dissipation of about 15mW per state, and requires 5V supply rails for proper operation due to the much level shifting occurring in the ACSU. Furthermore, it requires linear matched resistors, which can be quite difficult to match in practice (especially in a decoder with a large number of states) and occupy a large area on the die, and also physical MOS capacitors. Some high-speed analogue Viterbi decoders and their applications are listed in Table 4.3, which have appeared recently in the literature.

In this chapter the current-mode approach is investigated as an alternative to the voltage-mode approach for realising the ACS block of general classes of Viterbi decoder. The proposed approach enables a better trade-off between speed, size and power dissipation, than the approach in [13].

Table 4.3 Analogue Viterbi decoders and their applications.

Data rate (Mbit/s)	Number of states	Technology	Area (mm^2)	Power Dissipation (mW)	Application	Ref.
50	4	ECL/PCB	N/A	Not known	Satellite Com.	[12]
100	4	ECL/PCB	N/A	Not known	Optical Com.	[89]
50	2	CMOS 2μm	3.24	89	PR4	[5]
72	2	1μm BiCMOS	Not known	Not known	PR4	[6]
200	2	0.5μm BiCMOS	Not known	45	PR4	[7]
200	2	0.8μm BiCMOS	0.5	30	PR4	[8]
240	8	0.8μm BiCMOS	Not known	Not known	EPR4	[9]
> 80	8	0.8μm BiCMOS	0.3*	15†	EPR4	[13]

* Analogue section only. †Per state.

4.2 Current-Mode Realisation of the ACS Block

As mentioned in Section 4.1.3 the mathematical operations performed in the ACS block of a Viterbi decoder (and also a MFD) are well suited to implementation in the analogue domain since high accuracy is not required. Furthermore, if for the realisation of this subsystem analogue current-mode circuit techniques are employed, further savings in size and power dissipation compared to voltage-mode and digital realisations are possible. For example, 36 digital gates are required to perform a 4-bit summation, whereas two simple current sources perform the same function in the analogue current-mode domain. Therefore, the possibility for implementation using current-mode circuit techniques is very promising and is pursued here.

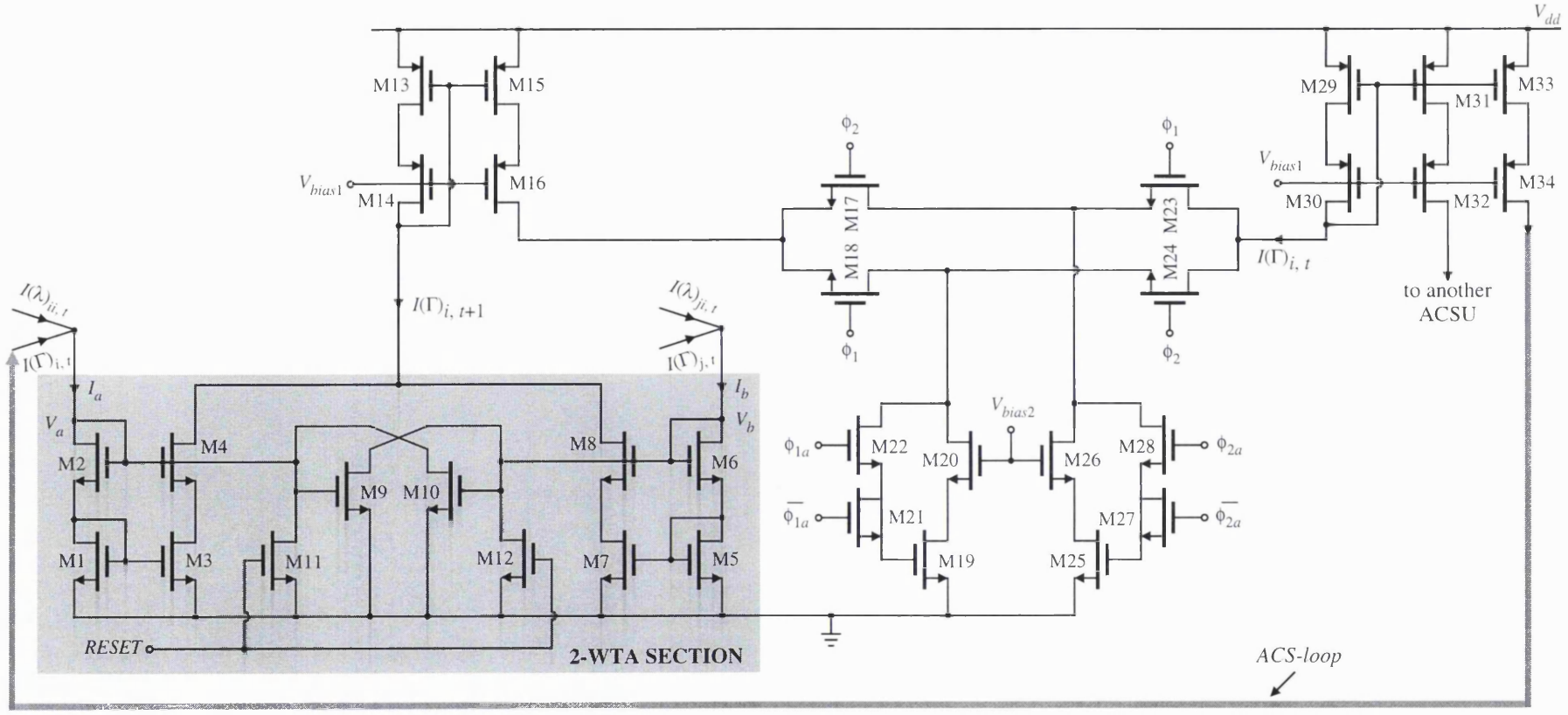


Figure 4.3 . A CMOS current-mode 2-input ACSU.

4.2.1 Circuit Description

The proposed circuit for realising the ACSU in Fig. 3.4b is the current-mode circuit shown in Fig. 4.3 where all branch and path metrics are in the form of currents and are represented by $I(\lambda)$ and $I(\Gamma)$, respectively. Only CMOS components are used in the circuit and unlike voltage-mode realisations [9], [13] no physical MOS capacitors or resistors are employed. Transistor dimensions are given in Table 4.4. The circuit consists of two sections. The first section, in the shaded box on the left of the figure, consists of a 2-input current-mode winner-take-all (2-WTA) circuit[1], which executes the path metric update (metric addition), compare and select functions indicated in Fig. 3.4b. The other section performs the metric storage and feedback functions indicated in Fig. 3.4b.

Table 4.4 Transistor dimensions for the ACSU in Fig. 4.3.

Transistor	Dimensions (μm)	
	W	L
M1-M8, M11, M12	8	0.8
M9, M10	12	0.8
M13-M16, M29-M34	40	0.8
M17, M18, M20, M23, M24, M26	20	0.8
M19, M25	20	1.6
M21, M27	2	0.8
M22, M28	4	0.8

The operation of the circuit is as follows. The two competing input metric currents are $I_a = I(\lambda_{ii,t}) + I(\Gamma_{i,t})$ and $I_b = I(\lambda_{ji,t}) + I(\Gamma_{j,t})$, and these are computed at the input of the two NMOS cascode current mirrors (M1-M4 and M5-M8). The competition phase of the ACS cycle is initiated by the *RESET* line going low turning off the NMOS transistors M11 and M12 (relevant timing waveforms are shown in Fig. 4.4). In addition to generating replicas of the currents I_a and I_b, the NMOS current mirrors realise voltages V_a and V_b at the drains of M2 and M6, respectively. These diode voltages operate the cross-coupled NMOS latch transistors M9 and M10 connected in a positive feedback loop. So, for example, if I_a is greater than I_b, then V_a is greater than V_b and the drain current of M9 is greater than that of M10. Since the drains of M9 and M10 are cross-coupled to the current summing nodes at the drains of M6 and M2 respectively, the drain currents of M6 and M5 decrease more rapidly than that of M2 and M1 until the smaller

[1] Comparison with other current-mode 2-WTA circuits from the literature is addressed in Chapter 5.

current is forced to zero and I_a "wins" the competition. In the case when I_a and I_b are equal, the circuit chooses randomly one of them as the "winner".

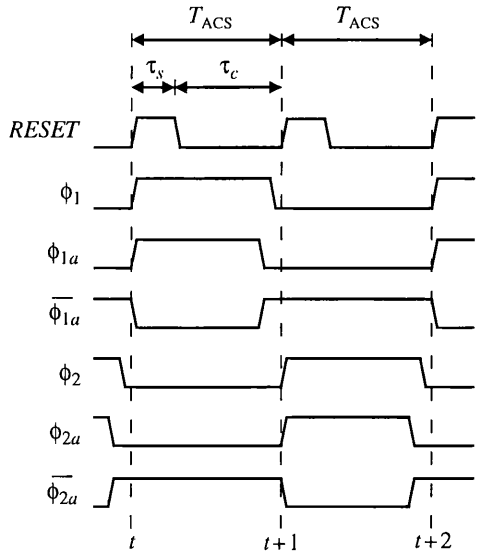


Figure 4.4 Timing waveforms for the ACSU in Fig. 4.3.

Having eliminated the smaller of the two input metric currents I_a and I_b, the larger ($I_{\text{in(max)}}$) is mirrored by the relevant NMOS 2-WTA mirror transistors and by the high swing PMOS current mirror (M13-M16), which forms the input to the metric storage unit. Note that this current is the new path metric current $I(\Gamma)_{i,\,t+1}$. To avoid the destructive effects of the stored path metric currents, two-stage current S/H cells are employed. The two S/H cells employed in this circuit are cascode *switched-current* (SI) memory cells (M19-M22 and M25-M28), and these operate in a "ping-pong" fashion. That is, one SI cell provides the previous path metric current to the ACS operation, while the other one stores the new path metric current. At the end of each ACS cycle their roles are interchanged (relevant timing waveforms are shown in Fig. 4.4). This arrangement was preferred over master-slave techniques [5] as it provides the potential of doubling the speed of the ACS cycle. In this circuit the stored path metric currents are feed back by means of the high-swing PMOS mirror M29-M34. Extending the output transistors of this mirror provides additional replications of the feed back current, which can be used for path metric normalisation. The use of SI techniques instead of *switched-capacitor* (SC) techniques eliminates the need for physical MOS capacitors and output buffers (source followers), and thus provides a more elegant and economical solution than voltage-mode

realisations of the ACS block [9], [13]. An introduction into the switched-current approach is given in Appendix 4.1.

Although the original version of the circuit when reported in [86] employed S^2I current memory cells [90] for combating *clock-feedthrough* (CFT) errors, further investigation revealed that simple cascode cells equipped with *dummy* switches (M21 and M27) is adequate for the 6-bits accuracy required for the ACS block. This simplification reduces the ACS cycle time, the circuit complexity and power dissipation.

Note that the complementary voltages V_a and V_b can be directly used to update the contents of SSM block without the need for extra comparison as is required in other analogue implementations [13]. This makes the circuit realisation of the ACS block very economical in terms of transistor count and power dissipation. Memory management implementation for Viterbi decoders is not investigated in this thesis.

Finally, the ACS cycle is terminated by the *RESET* line going high turning off the NMOS transistors M11 and M12. During the period when the *RESET* line is high computation of the metric currents I_a and I_b is performed. This period of time should be sufficiently large to allow for the branch metric currents to be computed.

4.2.2 Simulation Results

The propagation delay around the ACS loop (i.e., ACS cycle period) T_{ACS}, is given by

$$T_{\text{ACS}} = \tau_s + \tau_c \tag{4-1}$$

where,

τ_s is a safety factor to allow for the *RESET* process and for I_a and I_b to be computed.

τ_c is the settling time (to 0.1% accuracy) of the 2-WTA cascaded with a SI cell.

In addition τ_c may be further decomposed into the sum of the settling times of the 2-WTA ($\tau_{2\text{-WTA}}$) and current memory cell (τ_{SI}), as follows:

$$\tau_c = {}_{\max}(\tau_{2\text{-WTA}} + \tau_{\text{SI}}) \tag{4-2}$$

where the subscript $_{\max}$ refers to the slowest computation at each cycle corresponding to the smallest incident metric current.

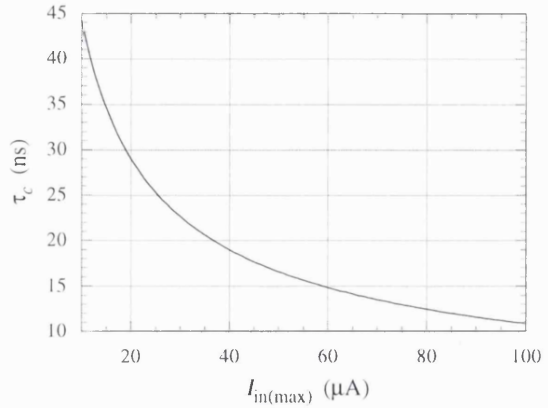


Figure 4.5 Operating speed performance of the ACSU in Fig. 4.3.

The circuit in Fig. 4.3 was realised using the AMS 0.8μm CMOS process with 5V supplies and SPECTRE parameters supplied by EUROPRACTICE [91]. Figure 4.5 shows values of τ_c as a function of $I_{\text{in(max)}}$ derived from simulations where it is observed that the operating speed of the ACSU depends on the magnitude of the largest input metric current. Further simulations revealed that the circuit could resolve input current differences $\Delta I = |I_a - I_b|$ of less than 1μA with negligible loss of operating speed. This corresponds to an accuracy of better than 6 bits over the entire dynamic range of 150μA that the ACSU has for 5V supply rails.

In order to calculate T_{ACS} the ACSU in Fig. 4.3 was simulated in recursive mode with τ_s of 5ns. The results of this simulation were an exactly 5ns level shifted version of the results in Fig. 4.5. The potential operating speed of an analogue Viterbi decoder employing this ACSU is determined by the value of the smallest metric current and the choice of this current must be traded-off against power dissipation. For example, if the minimum metric current is 20μA then from Fig. 4.5 $\tau_c = 29\text{ns}$ and so T_{ACS} is 34ns. Thus, the decoder should operate at about 30Mbit/s.

Finally in Fig. 4.6 the normalised results for the gain around the ACS loop are shown. From this graph it can be seen that the deviation from unity caused by analogue errors [12], mainly CFT errors and current mirror output conductance errors, is no worse than of 0.01% for the entire dynamic range of the ACSU. Thus, the 6-bits required accuracy is still maintained.

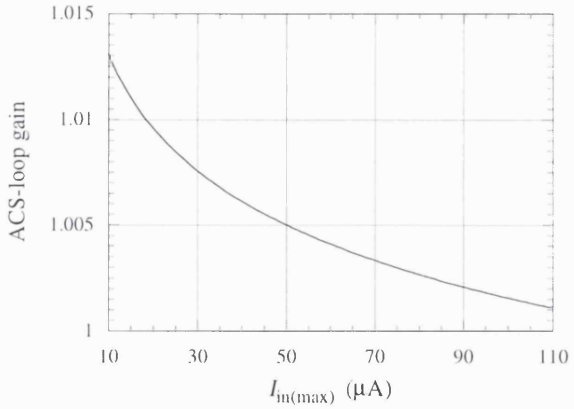


Figure 4.6 Deviation from unity gain.

4.3 ACSU Circuit Enhancements

The CMOS ACSU circuit described in Section 4.2.1, although very economical in terms of transistor count requires 5V supply rails for proper operation and therefore is unattractive for many modern applications. This voltage requirement is a direct consequence of the type of input NMOS current mirrors (M1-M4 and M5-M8) employed in Fig. 4.3 which result in V_a and V_b values of up to 3.2 V . In the following sub-sections *three* modifications to the 2-WTA section of ACSU in Fig. 4.3 are proposed [92] which not only enable the supply rails to be reduced from 5V to 2.8V without any loss of accuracy or dynamic range, but also reduce the ACS cycle period.

4.3.1 Circuit 1

In order to reduce the supply rails of the CMOS ACSU in Fig. 4.3 the obvious solution is to replace the input NMOS current mirrors (M1-M4 and M5-M8) of the 2-WTA section with other types of current mirror suitable for low-voltage applications. However, the current mirror selected must not degrade the operating speed or the accuracy of the ACSU. The high-swing cascode current mirror is a good candidate for low-voltage applications and has a similar high-frequency performance and accuracy to a simple cascode mirror [93]. Although replacing the cascode mirrors with high-swing mirrors reduces the supply rails of the ACSU, the current drive capability of the latch transistors M9 and M10 would be limited, adversely affecting the ACS cycle time. This is because the resulting voltages V_a and V_b would be in the order of only one diode drop which is

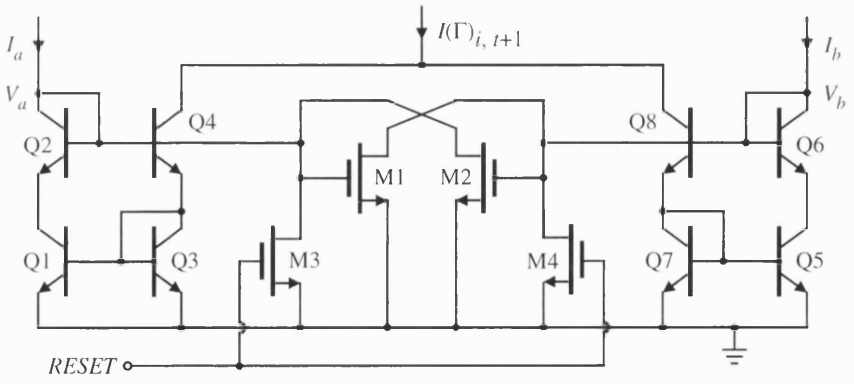


Figure 4.7 The 2-WTA section of circuit 1.

not sufficient to force rapid convergence of the current-steering latch in the clockwise or anticlockwise direction [94].

A better solution to this problem is to replace these mirrors by *npn* Wilson current mirrors[2], and the resulting BiCMOS 2-WTA circuit is shown in Fig. 4.7. The NMOS transistors M1-M4 in Fig. 4.7 have the same dimensions as those numbered M9-M12 in the 2-WTA section of Fig. 4.3. All *npn* transistors used in this circuit have the minimum size allowed in the AMS 0.8μm BiCMOS process. The use of bipolar rather than CMOS mirrors increases the speed of the 2-WTA by about 25% (see Fig. 4.8) and enables the supply rails to be reduced from 5V to 2.8V without loss of accuracy or dynamic range.

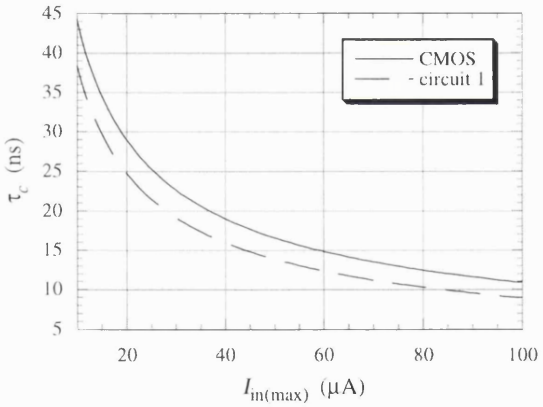


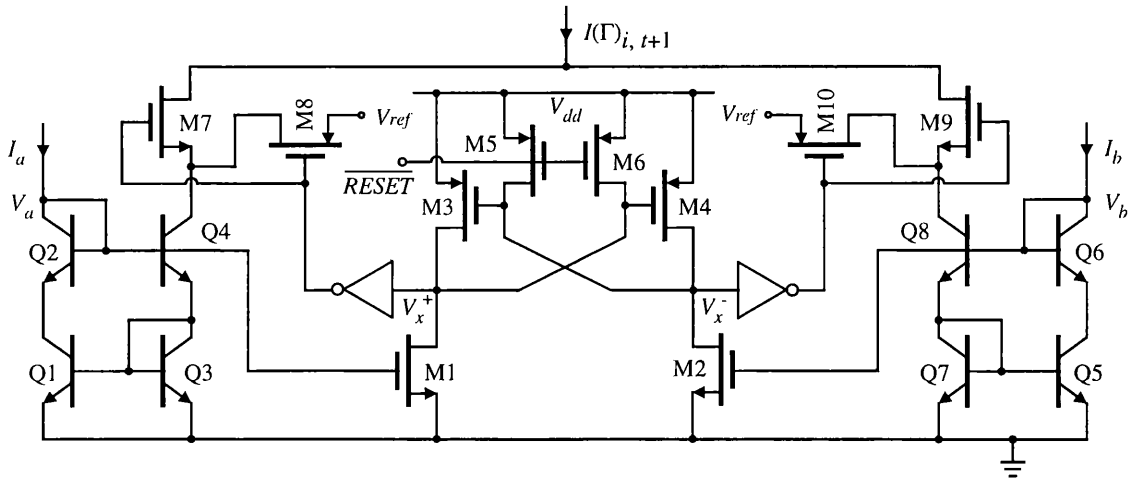
Figure 4.8 Comparison of the operating speed of the CMOS ACSU in Fig. 4.3 and circuit 1.

[2] The Wilson current mirror is preferred in bipolar realisations because the cascode mirror exhibits large errors, due to the fact that the base currents of all transistors are supplied by the input current only.

4.3.2 Circuit 2

An examination of the simulation results in Fig. 4.8 reveals that there is a noticeable loss in the operating speed of both ACSU's for low metric currents. In consequence, to maintain high-speed operation (e.g., above 50Mbit/s) the minimum metric current should be about 50μA at all times. On the other hand, it is desirable that the value of the smallest allowed metric current be as small as possible to minimise the overall power dissipation.

A closer examination of the operation of the 2-WTA's in Figs 4.3 and 4.7 reveals that the main factor limiting the speed of the ACSU employing these circuits is the need to turn their input current mirrors at the commencement of the competition phase of the ACS cycle. This is a direct consequence of the reset mechanism employed in both 2-WTA circuits, which pulls the inputs of the current mirrors to ground upon completion of the previous ACS cycle and discharges all the associated parasitic capacitance. It follows that the loss in operating speed, which results from this, is particularly noticeable for low input currents, as is confirmed by the simulation results in Fig. 4.8.



Transistor	M1, M2	M3 - M6	M7, M9	M8, M10	M11 - M14
W (μm)	5	15	20	15	40
L (μm)	0.8	0.8	0.8	0.8	0.8

Figure 4.9 The 2-WTA section of circuit 2.

In order to overcome this problem, the 2-WTA circuit shown in Fig 4.9 is proposed. In this circuit, the outputs of the *npn* input mirrors are connected to the PMOS high-swing mirror (M13-M16) in Fig. 4.3 by means of the *series* NMOS switches M9 and M11. These switches are operated by a pair of minimum geometry CMOS inverters shown

symbolically in Fig. 4.9, and by the PMOS current-sensing latch consisting of M3 and M4. The PMOS latch is driven by the NMOS current source transistors M1 and M2, which are driven by the voltages V_a and V_b developed across the input of the *npn* current mirrors (Q1-Q4 and Q5-Q7). The latch is reset by means of the PMOS transistors M5 and M6, which pull the inputs of the latch to V_{dd} at the beginning of the competition phase of the ACS cycle.

The PMOS switches M8 and M10 are complementary to M7 and M9 and ensure that the collector voltages of Q4 and Q8 are defined throughout the ACS cycle. This guarantees correct operation of the *npn* current mirrors. The voltage V_{ref} is arranged to be close to that at the input to the output PMOS mirror (M13-M16) in Fig. 4.3 to ensure that voltage variations across M7 (or M9) are kept at minimum, and therefore no significant loss of operating speed occurs from this effect. Conveniently, V_{ref} is the same as that required to bias M20 and M26 in Fig. 4.3 (V_{bias2}). The improvement in operating speed especially, for small values of metric current, is apparent in the simulation results shown in Fig. 4.10.

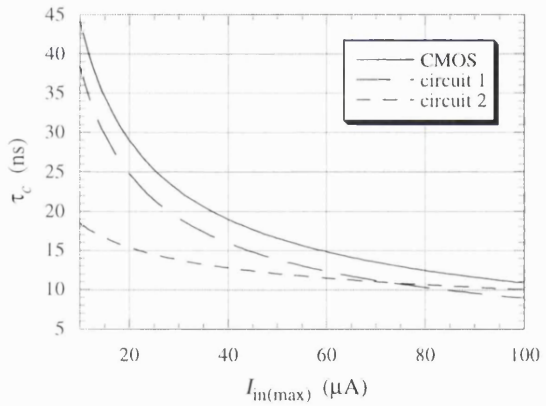
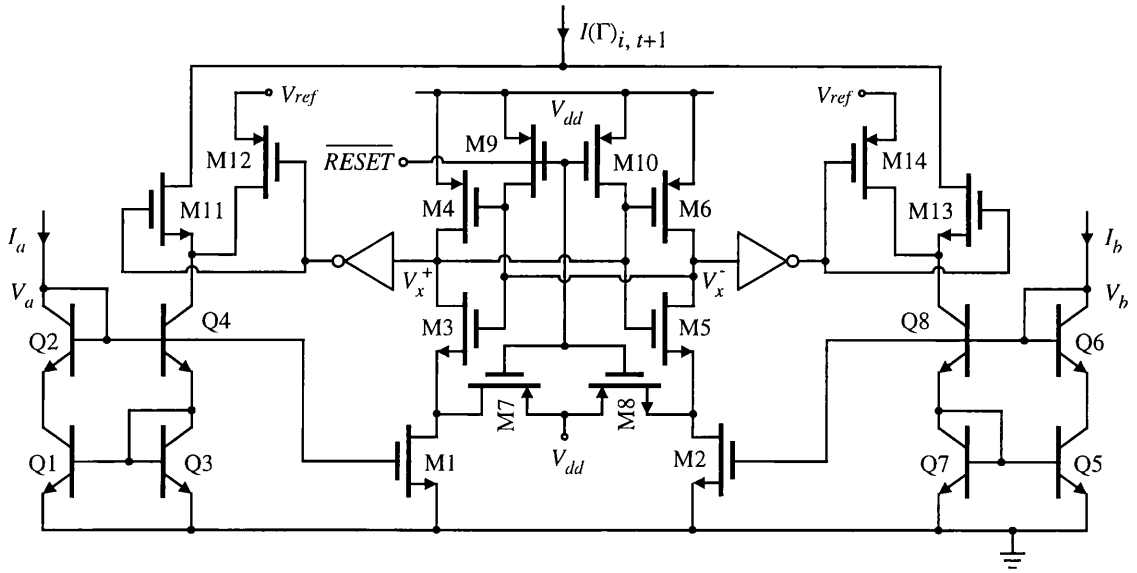


Figure 4.10 Comparison of the operating speed of the CMOS ACSU in Fig. 4.3, circuit 1 and circuit 2.

4.3.3 Circuit 3

Although the 2-WTA circuit in Fig. 4.9 enhances the operating speed of the ACSU for low metric currents, the circuit requires that the current source transistor (M1 or M2) corresponding to the largest input draws current throughout the whole ACS cycle. The other current source transistor only draws current during the period the reset transistors



Transistor	M1, M2	M3, M5	M4, M6	M7 - M10	M11, M13	M12, M14	M15 - M18
W (μm)	5	2	3.5	15	20	15	40
L (μm)	0.8	0.8	0.8	0.8	0.8	0.8	0.8

Figure 4.11 The 2-WTA section of circuit 3.

M5 and M6 are on. Consequently, the power dissipation of this circuit is about twice that of the 2-WTA circuit in Fig. 4.7, for the same input metric currents.

A solution to this problem is to replace the PMOS current-sensing latch in Fig. 4.9 with a CMOS current-sensing latch [95] as is shown in Fig. 4.11. The CMOS current-sensing latch consisting of M3-M6 is driven by the NMOS current source transistors M1 and M2, which are in turn driven by the voltages V_a and V_b developed across the input of the *npn* current mirrors as in Fig. 4.9. The CMOS current-sensing latch is reset by means of four PMOS transistors M7-M10, which pull the its inputs and outputs to V_{dd} upon completion of the ACS cycle. It is necessary to reset the inputs of the CMOS latch in addition to its outputs since this has the effect of driving M1 and M2 into deep saturation before the beginning of the competition phase of the ACS cycle. Note that by using the CMOS latch arrangement, M1 and M2 only draw current during the period M3-M6 are on and for the short time period required by the CMOS latch to make a decision. This minimises the power dissipation compared to the circuit in Fig. 4.9. Values of τ_c as a function of $I_{\text{in(max)}}$ for all four ACSU's described in this chapter are shown plotted in Fig. 4.12. The small loss in operating speed of circuit 4 compared to circuit 3 can be eliminated by further optimisation of the CMOS current-sensing latch. The main parameters of all

ACSU's are summarised in Table 4.5, together with those of [13] for comparison purposes.

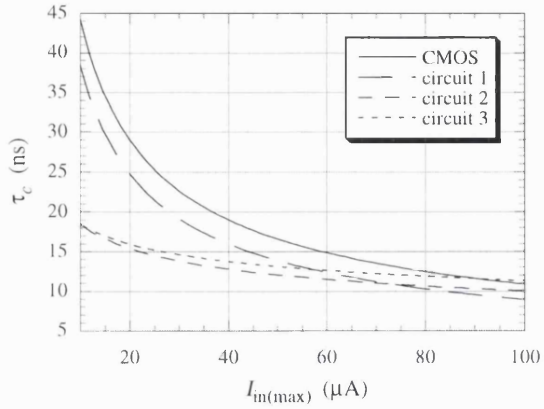


Figure 4.12 Comparison of the operating speed of the CMOS ACSU in Fig. 4.3, circuit 1, circuit 2 and circuit 3.

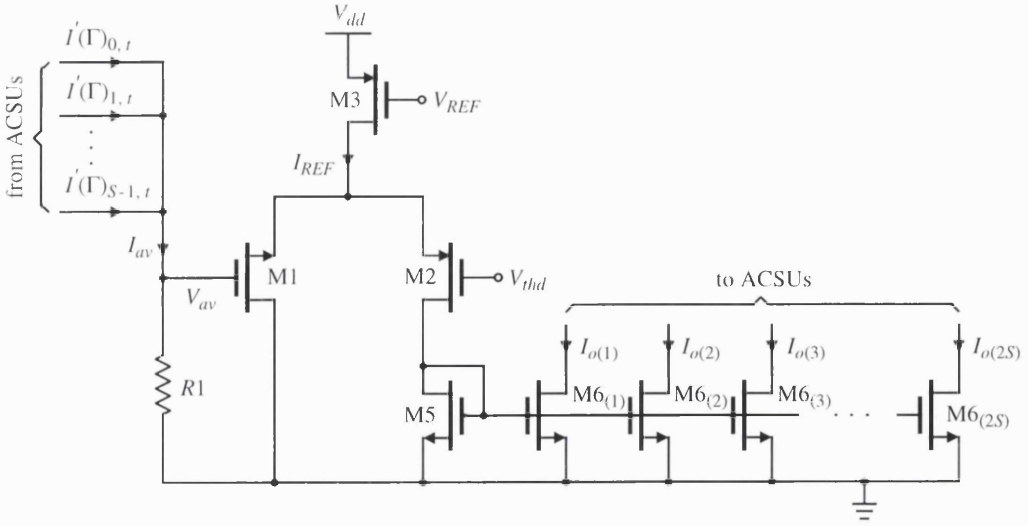
Table 4.5 Principal parameters of the ACSU's.

Parameter	ACSU				
	CMOS	Circuit 1	Circuit 2	Circuit 3	in [15]
Supply (V)	5	2.8	2.8	2.8	5
Active Area (mm^2)	0.007	0.008	0.009	0.01	0.03
T_{ACS} (ns)*	34	30	20	21	N/A
Decoder speed (Mbit/s)	29*	33*	50*	48*	> 80
Power dissipation (mW)	1.7†	0.95†	1.2†	1†	15

*When both metric currents are 20μA. †@ 50MHz.

4.4 Normalisation Circuitry

As seen from Eq. (3.4), unbounded growth of the path metrics is an inherent problem with the VA. To avoid path metric overflow and to reduce the required dynamic range in the ACSU, the averaging technique discussed in Section 3.2.2 is employed since it lends itself especially well to analogue realisation. To implement this normalisation technique, the circuit shown in Fig. 4.13 is proposed. This circuit operates on all incident path metric currents $I(\Gamma)_{i,t}$ by continuously monitoring these currents and maintaining a constant common-level for them. Note that the circuit contains no significant high-impedance nodes that can cause high-frequency instability problems when connected as a closed loop system within the ACSU.



Transistor	M1, M2	M3	M5, M6
W (μm)	15	6	8
L (μm)	0.8	1.2	1.2

Figure 4.13 The normalisation circuit.

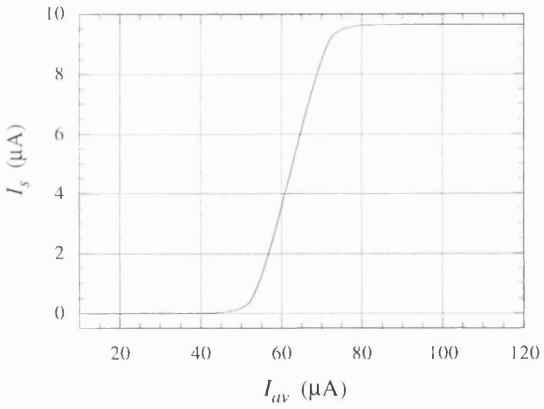


Figure 4.14 Typical DC response of the circuit in Fig. 4.13.

A copy of the path metric current from each ACSU can be obtained by extending the output transistors of the PMOS mirror (M29-M34) in Fig. 4.3. By arranging the width of the two added transistors to be S times smaller than the width of M29-M24, the sum of the resulting scaled path metric currents $(I'(\Gamma)_{0,t}, I'(\Gamma)_{1,t}, \ldots, I'(\Gamma)_{S-1,t})$ flowing through the n-well resistor $R1$[3] in Fig. 4.12 is the average I_{av} of all path metric currents. This current generates a proportional average voltage $V_{av} = I_{av}R$, which in turn is compared with the threshold voltage V_{thd} applied to the other input of the PMOS differential pair

[3] The value of this resistor is about 15K, and since no matching or precision is required it can be realised using the n-well layer of the processes for minimum layout area.

(M1-M2), in Fig 4.13. If V_{av} is greater than V_{thd} a proportion of the tail current I_{REF} is steered into the extended NMOS mirror (M5-M$6_l, l = 1, 0, .., 2S$). The resulting output currents $(I_{o(1)}, I_{o(2)}, \ldots, I_{o(2S)})$ are feed back to the input mirrors to each 2-WTA where they are subtracted from the existing path metric currents. The value of I_{av} can be controlled by the externally defined voltages V_{thd} and V_{REF}. A typical DC response of the circuit in Fig. 4.13 for which $I_{REF} = 10\mu A$ and $V_{thd} = 1V$ is shown in Fig. 4.14.

4.5 A Multiple Input ACSU for the MFDA

Although the ACSU described in Section 4.3 (or any of its variants in Section 4.4) would be adequate if employed for the realisation of the MFDA, the operating speed of the resulting decoder would be rather low compared to a Viterbi decoder. This is a direct consequence of the number of sequential steps involved in each decoding cycle of the MFDA as explained in Section 3.6. For example, consider the design of a $R = 1/2$ MFD with $K = 5$ and $W = 5K$ (the selection of W is according to the simulation results in Section 3.6.1). Assuming that the parallel part of the decoder including BMC block update and output WTA network classification requires about 15ns, the decoding period of a MFD, T_{MFD} (expressed in ns) is approximately

$$T_{\text{MFD}} = \{T_{\text{ACS}} \cdot (W - K)\} + 15 \qquad (4\text{-}3)$$

Given that $T_{\text{ACS}} = 15\text{ns}$, then according to Eq. (4-3) the MFD under consideration would operate at about 3Mbit/s. On the other hand, the equivalent analogue Viterbi decoder realised using the same ACSU would operate at about 66Mbit/s. This emphasises the need for a multiple input ACSU so that the *N*-step trellis technique described in Section 3.6.2.2 can be applied to the realisation of the MFDA, to reduce the number of sequential steps, and thus maximise the achievable decoding speed.

Although the 2-WTA section of the ACSU in Fig. 4.3 can be easily modified to accommodate more inputs, the circuit complexity would increase as a function of M^2, where M the number of inputs is 2^N. A better solution is to employ a current-mode WTA circuit whose complexity increases linearly with M, such as the one shown in Fig. 4.15 [96]. In the circuit in Fig. 4.15, the input metric currents $I_1, I_2, \ldots, I_M$ are converted into gate-source voltages $V_1, V_2, \ldots, V_M$ at the inputs of the NMOS diode connected transistors M1-MM. These voltages are then applied to the base terminals of the bipolar transistors Q1, Q2, ..., QM connected as voltage followers with a common output node

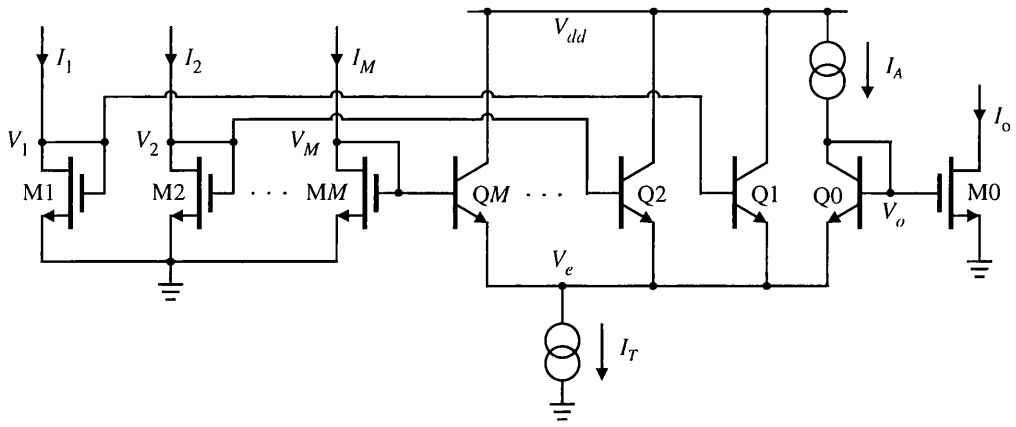
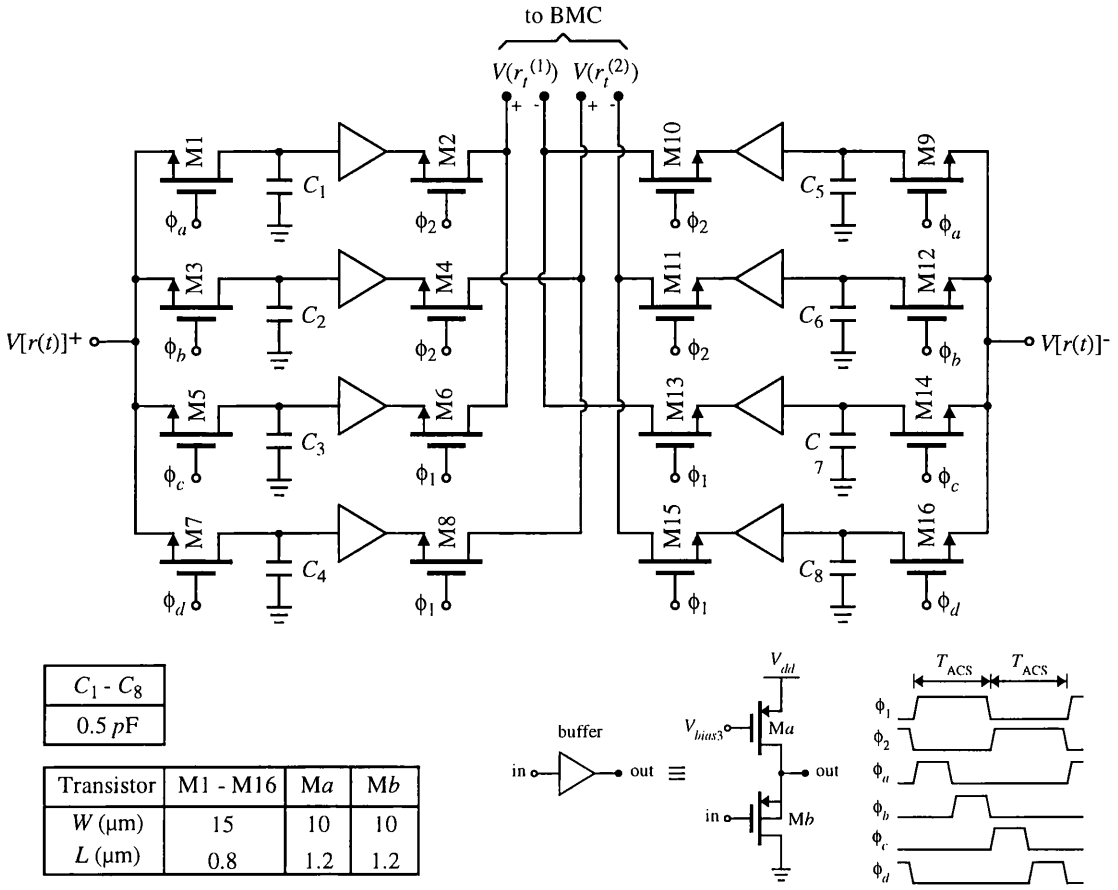


Figure 4.15 A multiple input WTA circuit.

that carries voltage V_e. The value of V_e follows the maximum of $V_1, V_2, \ldots, V_M$ with a DC level shift of approximately one diode drop. The output bipolar transistor Q0 and the current source I_A whose value is about half that of the tail current source I_T compensates for the DC level shift in the output voltage V_o. In addition to the diode drop in the value of V_e, there is an additional offset error due to the non-ideality of the diode characteristics (Q1, Q2, ..., QM) [13]. The magnitude of this error depends on how closely spaced the values of $V_1, V_2, \ldots, V_M$ are. Techniques for cancelling or minimising this error have been proposed [97], but at the expense of reducing the operating speed of the circuit. Finally, V_o is converted into an output current I_o (corresponding to the largest of $I_1, I_2, \ldots, I_M$) by means of the NMOS transistor M0. It should be noted that the circuit in Fig. 4.15 belongs to a class of WTA networks known as the *current-conveyor* [98]. The current-conveyor WTA approach is further discussed in Chapter 5.

Preliminary simulation results indicated that when a cascode 8-input version of the circuit in Fig. 4.15 (i.e., all NMOS transistors are cascoded) implements the WTA section of the ACSU in Fig. 4.3 (with 2.8V supply rails), T_{ACS} values of about 10ns are possible. Thus, for the MFD example considered above, an 8-input ACSU employing this type of WTA network would reduce the number of sequential steps by a factor of 3 and the T_{MFD} from 315ns to 115ns. This translates to an operating speed of about 9Mbit/s as opposed to 3Mbit/s obtained earlier without the 3-step trellis approach. Further increase in operating speed is possible by using a WTA with a larger number of inputs.



Transistor	M1 - M16	Ma	Mb
W (μm)	15	10	10
L (μm)	0.8	1.2	1.2

Figure 4.16 Front-end circuitry for an analogue BPSK Viterbi decoder.

4.6 Interface Circuitry

In most analogue communication circuits it is desirable to keep the signals coming into the system in differential form instead of single-ended form to combat destructive effects such as common-mode noise and CFT errors [99]. Furthermore, in low-voltage applications the use of differential signals is usually necessary to enable large input signal swings. In an analogue convolutional decoder design the input signal could be either in current or voltage form. In this case, the representation of the input signal in voltage form was selected because it simplifies the realisation of the BMC block and also keeps the power dissipation of the front-end circuitry in both types of decoder to the minimum.

4.6.1 Viterbi Decoder Interface

An efficient realisation for the interface circuitry of a soft analogue BPSK Viterbi decoder is shown in Fig. 4.16 where all signals are fully differential. The circuit operates with 2.8V supply rails and is designed to accommodate nominal input voltage swings of up to

1V peak-to-peak centred at 0V. The first row of S/H elements in Fig. 4.16 operates in a "ping-pong" mode with the third row of S/H elements storing the odd channel symbol samples. The even channel symbol samples are stored by the second and fourth row of S/H elements, which also operate in a "ping-pong" fashion. The relevant timing waveforms are shown in Fig. 4.16. Note that the clock waveforms ϕ_1 and ϕ_2 are the same as in Fig. 4.4, but here are further subdivided into (ϕ_a, ϕ_b) and (ϕ_c, ϕ_d), respectively. The period of these four sub-clock waveforms is the same as that of ϕ_1 and ϕ_2 (i.e., $2T_{ACS}$ seconds). However, their on period is $T_{ACS}/2$ seconds, which corresponds to the bit rate. The samples placed on the eight capacitors are buffered with PMOS source followers, shown as triangles in the figure. The use of PMOS source followers provides the required level shifts to interface with the BMC block.

4.6.2 MFD Analogue Delay Line

As mentioned in Section 3.7.1 the MFDA requires an analogue delay line of which the simplest way to construct such a delay line is to use a serial architecture where a cascade of S/H elements forms the delay line. However, in this architecture errors such as offset and noise add up as the signal is transferred from one stage to another, and this limits the maximum possible number of elements in the cascade. Therefore, this approach is not particularly suitable for the implementation of the SS block of a MFD where the received sequence is required to be stored for a window of depth $W \approx 5K$. A better approach is to use a parallel architecture [100], where the S/H elements sequentially sample the input signal and then hold it for the following $W-1$ clock periods.

Figure 4.17 shows half of the block diagram (the one that interfaces to the positive input voltage line) of the SS block for a soft analogue BPSK MFD (the input voltage swings are the same as in Fig. 4.16). The SS block consists of two analogue tapped-delay lines each with W delay taps spaced at the bit rate, i.e., T_{MFD} seconds. The upper delay line holds the odd channel symbol samples while the lower delay line holds the even channel symbol samples. Note that a delay of $T_{MFD}/2$ seconds is inserted at the input of the upper delay line to time align the odd with the even channel symbols. The first tap in each delay line holds the sample at time t, the second tap holds the sample at time $t-1$, the third tap holds at time $t-2$, and so on, where each time decrement corresponds to one T_{MFD} period. A rotating switch with $2W$ switches controlled by $W-K+1$ non-overlapping clocks is used to connect the taps of the analogue delay lines with the BMC block in the

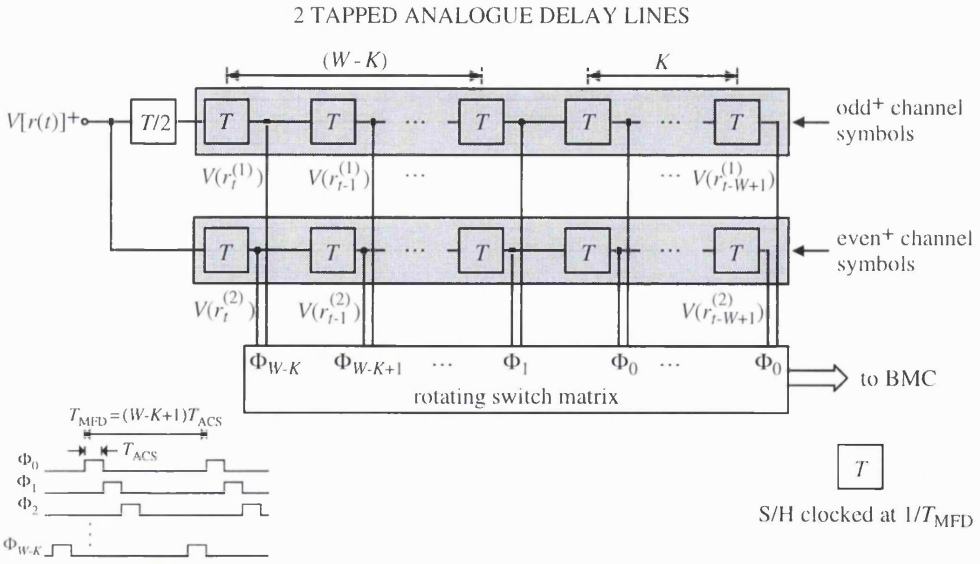


Figure 4.17 Block diagram of the SS block of an analogue BPSK MFD.

manner described in Section 3.7.1. The switch controlling waveforms are staggered as shown in Fig. 4.17 each with an on period of T_{ACS} seconds. These clock waveforms can easily be derived from the outputs of a digital ring counter with $W-K+1$ stages.

The schematic diagram for the lower analogue delay line in Fig. 4.17 is illustrated in Fig. 4.18 (the schematic for the upper delay line is the same). This circuit is composed of an array of $W+1$ S/H elements, a $W \times W$ rotating NMOS switch and a digital ring counter (not shown in the figure). The corresponding switch controlling waveforms are staggered as shown in Fig. 4.18 each with a period of $(W+1)T_{MFD}$ seconds. The buffers are implemented as PMOS source followers with the same transistor dimensions as in Fig. 4.16 and also provide the required level shifts to interface with the BMC block.

4.7 Branch Metric Generators

In most cases, the symbol metrics are expressed as linear combinations of the input sample and some DC values. As an example, Eq. (3-9) gives the symbol metrics for BPSK in the presence of AWGN where it is observed that only negation operations are required to compute the symbol metrics. In an efficient realisation, differential transconductors [15] can be used to convert the differential voltages from the output of the front-end circuitry to differential currents. The resulting current signals, with appropriate polarities can be combined to form the branch metrics by simply interconnecting the outputs of the tranconductors. Since high linearity is not required

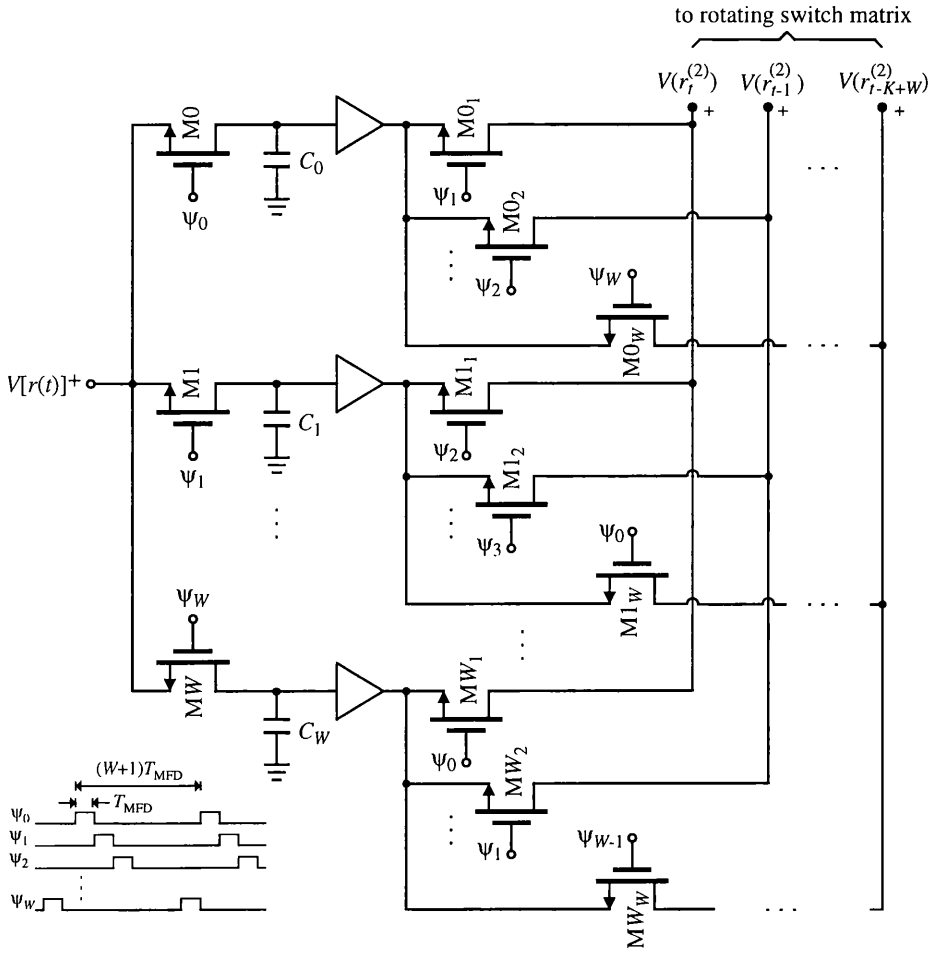
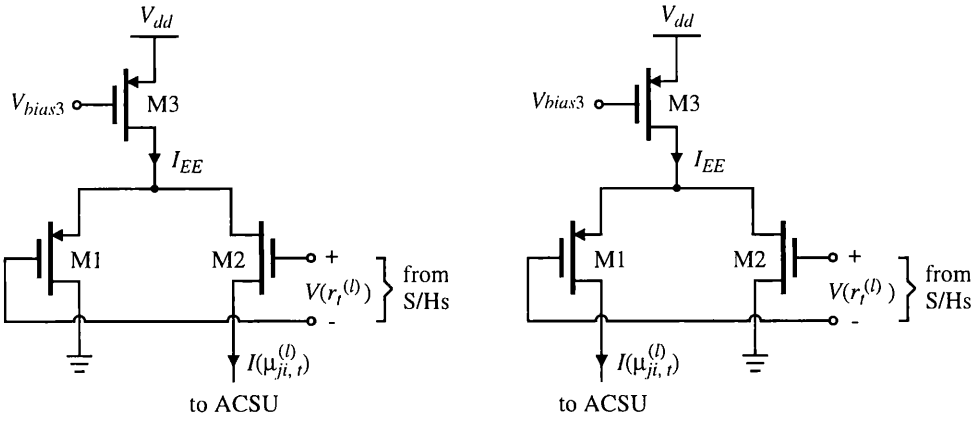


Figure 4.18 Circuit realisation of the tapped delay line in Fig. 4.17.



Transistor	M1, M2	M3
W (μm)	2.4	10
L (μm)	1.2	1.2

Figure 4.19 Symbol metric generators for analogue BPSK convolutional decoders.

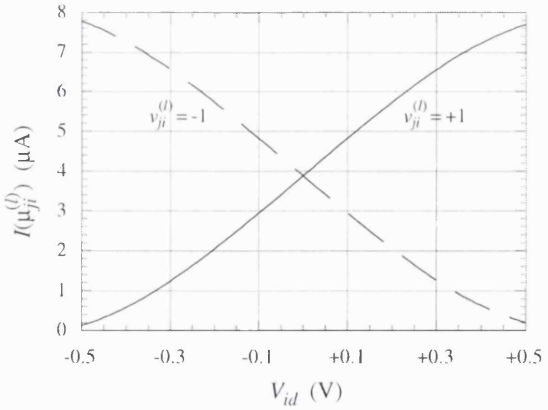
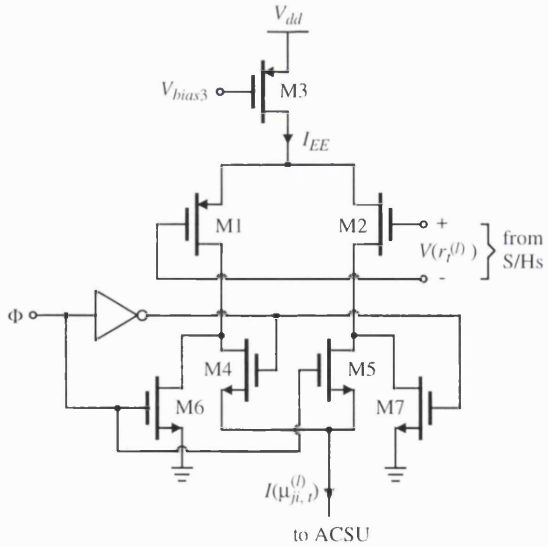


Figure 4.20 DC responses of the transconductors in Fig. 4.19.

(only about 3-bits of accuracy) simple PMOS differential pairs can be used because they can be directly interfaced with the current-mode ACSU's described in this chapter. Figure 4.19 shows the schematic diagram of the PMOS transconductor cells and Fig. 4.20 shows the DC response of these cells corresponding to the two cases in Eq. (3-9) when the tail current I_{EE} in Fig. 4.19 is set to 8μA. Finally, Fig. 4.21 shows a programmable transconductors that can be used for the variable branch symbols in the BMC of a MFD. The output of this transconductor is controlled by the clock pulse Φ whose state corresponds to the value of the variable branch symbol in consideration.



Transistor	M1, M2	M3	M4, M5	M6, M7
W (μm)	2.4	10	20	2
L (μm)	1.2	1.2	0.8	0.8

Figure 4.21 A programmable transconductor for use in MFD's.

Appendix

4A

Introduction to Switched-Current Systems

A *switched-current* (SI) system may be defined as a system using analogue sampled-data circuits in which signals are represented by current samples. This is in contrast to *switched-capacitor* (SC) circuits in which signals are represented by voltage samples. Unlike SC circuits, which require additional processing (second layer of polysilicon) to fabricate linear MOS capacitors, SI circuits can be integrated in an economical single-poly (digital) CMOS process. The only elements needed in SI circuits are PMOS and NMOS transistors, and the small gate-source parasitic capacitance C_{gs} of an MOS transistor is exploited to sample and hold charges. In addition, SI circuits have the potential for low voltage operation because their internal voltage swings need not be large since signals are represented by currents [16]. Other outstanding features of this technique include wide bandwidth, wide dynamic range and easy manipulation of signals. Because of its apparent overwhelming advantages, the SI technique has been recently receiving considerable attention in numerous application areas [101]-[103].

4A.1 Basic Switched-Current Cells

The two basic current sampling cells are the *current track-and hold* (T/H) and *current copier* cells.

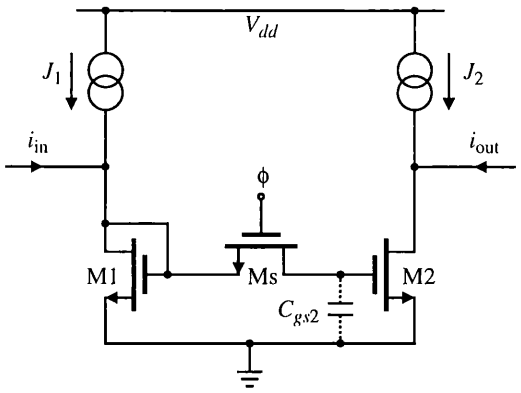


Figure 4A.1 Basic track-and-hold SI cell.

4A.1.1 Current Track-and-Hold Cell

The *current track-and-hold* (T/H) [104] cell is shown in Fig. 4A.1. It is constructed by placing a switch-transistor between the gates of the current mirror transistors M1 and M2. The two bias current sources J_1 and J_2 in Fig. 4A.1 are optional and serve to allow transmission of bipolar current signals. The added transistor, Ms, is switched on and off by a precise digital clock signal ϕ. When ϕ is pulsed high, Ms is turned on and the gates of the two current mirror transistors are shorted. In this state the circuit functions as a current mirror, and with an input current applied to the drain of M1, the output current i_{out} tracks the input current i_{in}. When ϕ is pulsed low, Ms is turned off, and the gates of M1 and M2 are disconnected. The gate voltage of M1, corresponding to the value of the input current at the instant Ms is turned off, is sampled into the small gate-source parasitic capacitance C_{gs2} of M2. While Ms is switched off, the gate-source voltage V_{gs2} of M2 remains constant; and therefore i_{out} is held at a constant value corresponding to i_{in} at the instant when Ms was turned off. Realistically, junction leakage currents in the switch-transistor Ms gradually discharge C_{gs2} over some time. The level of this non-ideal behaviour determines the minimum sampling frequency for a given accuracy [105].

4A.1.2 Current Copier Cell

The main drawback of the current T/H cell is that its accuracy depends on the precise matching of transistors M1 and M2 and the two bias current sources J_1 and J_2. The current copier [106] (or *dynamic current mirror*) eliminates this problem by employing a single transistor and a single bias source J for both input and output currents. The basic current copier cell is shown in Fig. 4A.2 together with the relevant clock waveforms.

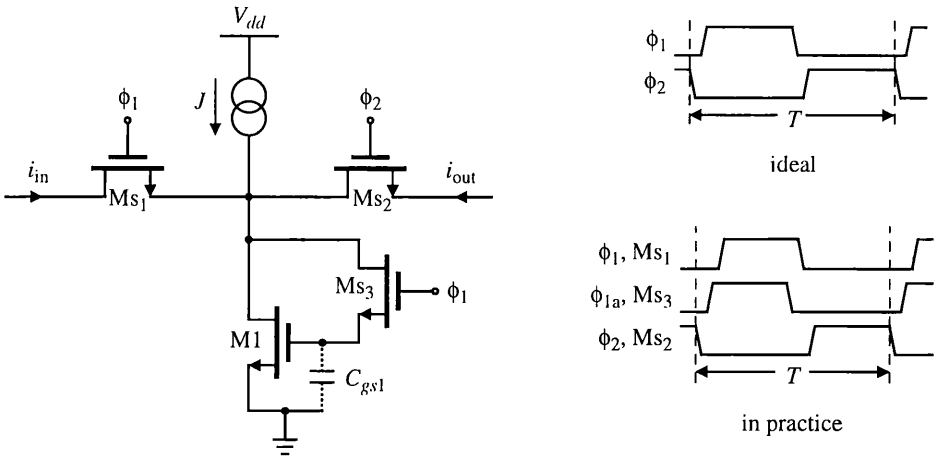


Figure 4A.2 Basic current copier and timing waveforms.

Two-phase non-overlapping clocks (ideal clock waveforms in Fig. 4A.2) control the transistor switches that configure the circuit first as a tracking amplifier and then as a holding amplifier. The operation of the circuit is as follows. When ϕ_1 is pulsed high (Ms_1 and Ms_3 turned on) and ϕ_2 is pulsed low (Ms_2 turned off), M1 is diode-connected and the total input current $J + i_{in}$ flows initially into the discharged gate-source parasitic capacitance C_{gs1} of M1. As C_{gs1} charges, the gate-source voltage V_{gs1} of M1 rises and when it exceeds the threshold voltage V_{TH}, M1 conducts. Eventually, when C_{gs1} is fully charged, $J + i_{in}$ flows in the drain of M1.

To configure the current copier as a hold amplifier, ϕ_2 is pulsed high (Ms_1 turned on) and ϕ_1 is pulsed low (Ms_1 and Ms_3 turned off). The value of V_{gs1} corresponding $J + i_{in}$ just before Ms_3 was turned off is held on C_{gs1}. With Ms_2 turned on, the imbalance between currents J and $J + i_{in}$, forces an output current, $i_{out} = -i_{in}$, to be sensed at the output. This current is a memorised version of i_{in} and is achieved by virtue of the charge retained on the gate-source parasitic capacitance C_{gs1}. In practice, the pulse driving Ms_3 must be slightly in advance of that driving Ms_1 to ensure that the input current has been properly memorised before it is interrupted (in practice clock waveforms in Fig. 4A.2).

Although the current copier cell eliminates the errors due to the transistor and bias source mismatches present in the current T/H cell, it results in a higher harmonic distortion than the current T/H cell [107]. This is a result of the large internal transient glitches that occur in the current copier cell.

4A.2 Accuracy Limitations in SI Circuits

The two basic current sampling cells do not have a great accuracy in practice. Their main limitations include *channel length modulation*, *charge injection*, *leakage currents*, and *noise* [16]. The level of these errors is such that SI systems have inadequate precision and linearity for many applications. Channel length modulation and charge injection errors are by far the most serious causes of errors and if not compensated for seriously degrades the performance of SI systems.

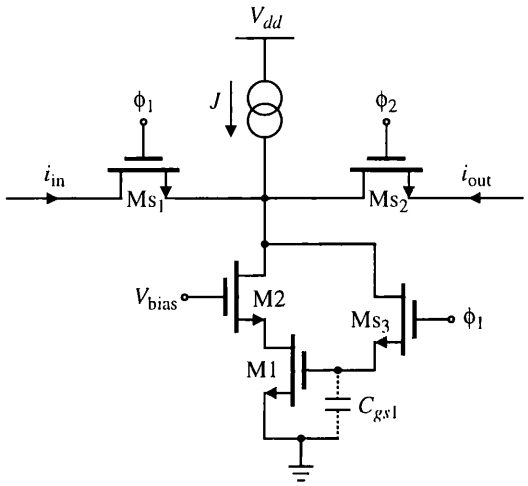


Figure 4A.3 Cascode SI cell.

4A.2.1 Channel Length Modulation Errors

Due to MOS channel length modulation, the circuit shown in Fig. 4A.2 will not work properly if the drain-source voltage V_{ds1} of M1 during phase ϕ_1 differs from that occurring during phase ϕ_2. This problem can be avoided by arranging for both ϕ_1 and ϕ_2 to be performed under constant V_{ds1} conditions (i.e., using a servo amplifier [106]). A more economical solution is to use cascoding [105] as shown in Fig. 4A.3. Cascoding reduces the variation in V_{ds1} by a factor approximately equal to $g_{m2}r_{ds2}$ where g_{m2} and r_{ds2} are the transconductance and output resistance of the cascode transistor M2, respectively. This factor is typically 100 and gives the cascoded memory cell a current gain error of less than 0.1% (i.e., 100 times lower than that of the basic current copier cell). When designing with this cell short settling times can easily be achieved by ensuring that the pole frequency associated with the source of M2 is at least an order of

magnitude higher than that of M1 [16]. This can be achieved by giving M2 a high aspect ratio (W_2/L_2).

An even lower current gain error of about 0.001% can be achieved with the regulated cascode memory cell [108] shown in Fig. 4A.4, which is based on the regulated cascode current mirror [108]. In this circuit the drain-source voltage V_{ds1} of M1 is regulated to a fixed value by the feedback loop consisting of an amplifier (M3 and bias current source I, with $I << J$) and a follower, M2. Thus, both phases ϕ_1 and ϕ_2 are performed under constant V_{ds1} conditions. When designing with this cell short settling times are difficult to achieve because the addition of the regulation amplifier makes the memory loop a third-order system [16].

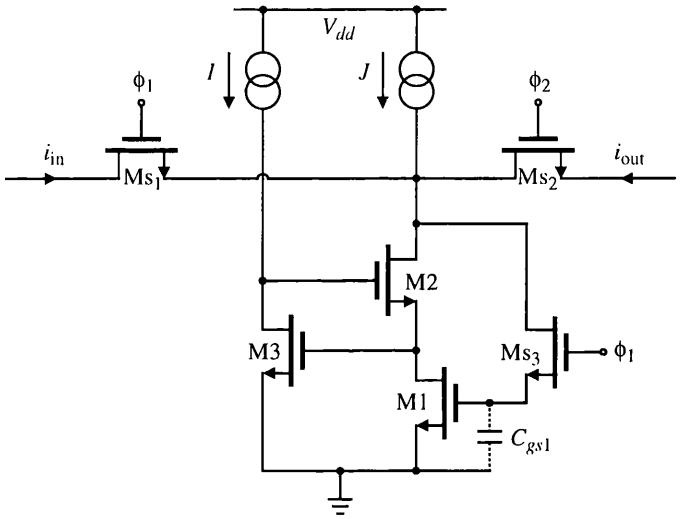


Figure 4A.4 Regulated cascode SI cell.

4A.2.2 Charge Injection Errors

CFT induced charge injection is the most serious cause of errors in SI circuits and it introduces a trade-off between speed and accuracy [105]. CFT originates from the charge injected from the channel capacitance, C_{ch} $(= 0.5C_{ox}W_{eff}L_{eff})$, and overlap capacitance, C_{ov}, of the MOS switch-transistor into the storage capacitor when the former is turned off. As Fig. 4A.5 illustrates, a portion Δq_1 of this charge is pushed into C_{gs1} and in turn changes V_{gs1} by [109]

$$\Delta V_{gs1} = \frac{\Delta q_1}{C_{gs1}} \tag{4A-1}$$

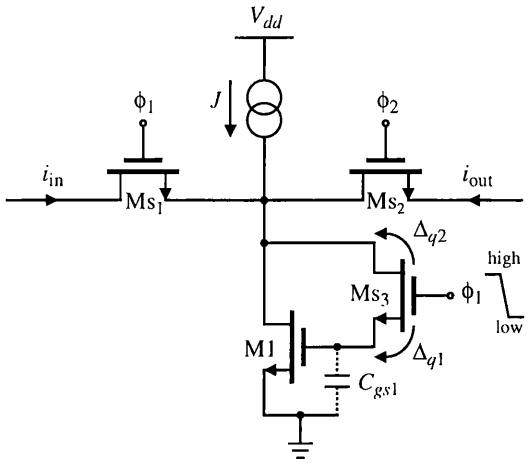


Figure 4A.5 Origins of CFT induced charge injection.

The variation in V_{gs1} results in a change of the drain current i_{ds1} of M1 (CFT error) by

$$\Delta i_{\text{CFT}} \approx g_{m1} \frac{\Delta q_1}{C_{gs1}} \tag{4A-2}$$

where g_{m1} is the transconductance of M1 and it equals to

$$g_{m1} = \sqrt{2\mu C_{ox} \frac{W_1}{L_1} i_{ds1}} = \frac{2 i_{ds1}}{V_{gs1} - V_{TH}} \tag{4A-3}$$

The charge Δq_1 injected into C_{gs1} depends on the switching speed, the source to substrate voltage, and the capacitance ratio between C_{gs1} and the switch-transistor total capacitances $(C_{ch} + C_{ov})$. The larger C_{gs1} and $V_{gs1} - V_{TH}$ are the smaller Δi_{CFT} and the better the accuracy will be. There are also other benefits from having large C_{gs1} and $V_{gs1} - V_{TH}$, namely high signal-to-noise ratio and low distortion. Unfortunately, this condition conflicts with high-speed operation because the ratio C_{gs1}/g_{m1} in Eq. (4A-2) is the settling time constant of the memory cell that determines the maximum sampling frequency [105]. For a given current signal level the fastest settling time is obtained for low values of $V_{gs1} - V_{TH}$ and C_{gs1} (i.e., a transistor with short channel length and a high g_{m1}). From Eq. (4A-3) it is noted that g_{m1} depends on the signal current being sampled and thus the CFT error is both *signal-independent* and *signal-dependent* [104]. The latter type of CFT error arises when the voltage at the terminals of the switch-transistor varies with the signal level [110].

To reduce charge injection errors in SI circuits several circuit techniques have been developed. The most common technique is the use of a dummy switch[4] [111] whose gate is controlled by an inverted clock signal as shown in Fig. 4A.6. The aim is to inject an equal and opposite charge into the storage capacitor and hence cancel most of the CFT error. This technique requires two clock phases, and generally its accuracy depends on the timing relationship between the clock phase edges.

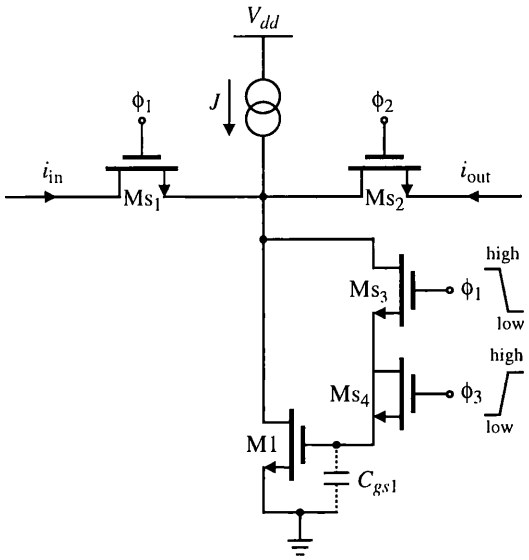


Figure 4A.6 The dummy switch compensation technique.

Fully-differential circuits are an alternative solution for rejecting CFT errors and errors due to even-order nonlinearities [99]. Other circuit techniques for reducing signal-independent and/or signal-dependant CFT errors include the replication technique [104], [112], high-gain negative feedback [110], Miller capacitance-enhancement [109] and two-step cancellation (S^2I) [90]. Of these techniques the S^2I approach [90] seems to offer the most economical solution in terms of silicon area, power dissipation and loss of operating speed.

Figure 4A.7 shows the S^2I cell and its clock waveforms. The operation of the circuit is as follows. When ϕ_{1a} is pulsed high, the coarse-memory transistor M1 is diode-connected and the fine-memory transistor M2 is connected to the bias voltage V_{ref}. During this phase a bias current J is generated in M2 and the total drain current in M1 is $J + i_{in}$. Next, ϕ_{1b} is pulsed high and M2 is now diode-connected while the gate of M1 floats. The

[4] A dummy switch is approximately one-half the size of the switch-transistor and has its drain and source terminals shorted.

input signal current i_{in} is still flowing into the input so that transistor M1 holds a current $J + i_{in} + i_{CFT}$ and the current in M2 is $J + i_{CFT}$ where i_{CFT} is the signal-dependant CFT error. Finally when ϕ_2 is pulsed high, the gate of M2 is opened and an extra error Δi occurs in M2 mainly due to charge injection. The current in M2 is $J + i_{CFT} + \Delta i$, the current in M1 is $J + i_{in} + i_{CFT}$, and the output current is $i_{out} = -i_{in} + \Delta i$. Therefore, a small almost constant offset remains at the output. The same principle is used in the auxiliary cell technique [109]. However, this approach requires a somewhat more complicated clocking scheme than the S^2I scheme.

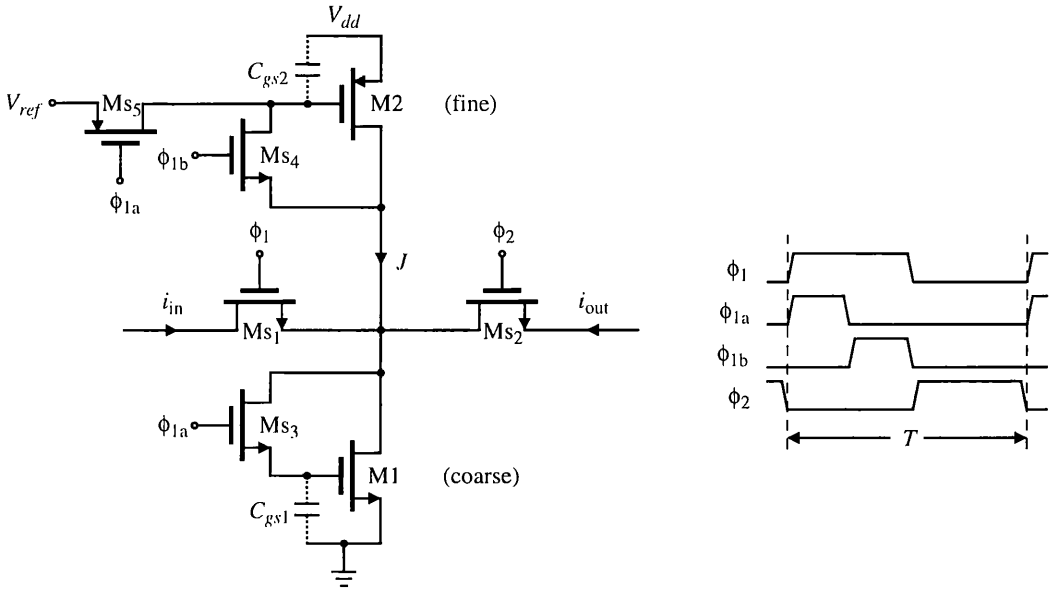


Figure 4A.7 The S^2I principle and timing waveforms.

4A.2.3 Leakage Currents

As Fig. 4A.8 illustrates, after the MOS switch-transistor Ms_3 is turned off, another effect influences the gate-source voltage V_{gs1} of M1. That is, a small current flows through the reverse-biased diode D (appears between the source and substrate of Ms_3), drawing charge from C_{gs1} continuously. Thus, over a time interval Δt this leakage current i_{leak} will change V_{gs1} by [113]

$$\Delta V_{gs1} = i_{leak} \frac{\Delta t}{C_{gs1}} \tag{4A-4}$$

with a resulting change in the drain current of M1 by

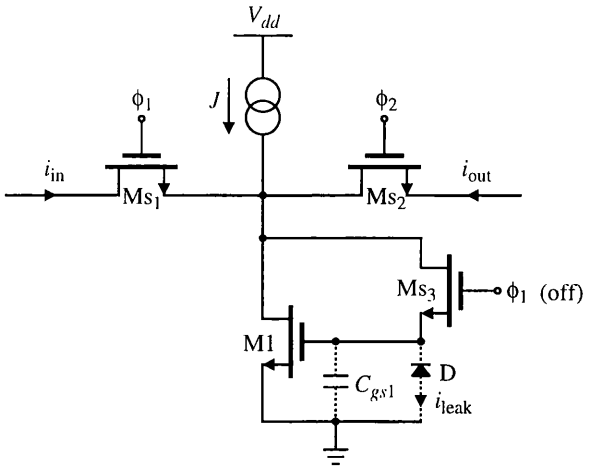


Figure 4A.8 Origins of leakage currents.

$$\Delta i_{ds1} = g_{m1}\Delta V_{gs1} \tag{4A-5}$$

The leakage current i_{leak} determines the minimum sampling rate of the memory cell for a given accuracy. Leakage currents are negligible with high-speed current memory cells but should be taken into consideration at low-speeds. Because the leakage currents increase rapidly with temperature, high accuracy is not possible with current memory cells at high temperatures (i.e., over $40\,^{o}C$) [16].

4A.2.4 Noise

Of the various types of noise present in analogue circuits, two are important in SI circuits: thermal noise and flicker $(1/f)$ noise. Thermal noise originates from the random motion of electrons and depends on temperature but not on current flow. On the other hand, flicker noise does not depend on temperature but is rather proportional to the current flow. It is believed to be a result of mobility variations. A detailed description of noise effects in SI systems is given in [16].

Chapter

5

An Analogue Winner-Take-All Network for Large-Scale Applications

THE *winner-take-all* (WTA) network [114] is an important component of many types of artificial neural network (ANN), of which the *classifier* is a typical example [115]. The function of a WTA network is to select and identify the largest (or maybe the smallest) variable (which can be binary or continuous-valued) from a specified set of M competing variables and inhibit the remaining $(M-1)$ variables. In a classifier, each of the M inputs to the WTA represents the level of correlation between the presented input pattern and one of the M stored memories. Potential applications for such systems include VQ for image compression [116]-[118] since a vector quantiser is very analogous to a classifier whose output is arranged to be a digital representation of the appropriate matching codeword [119]. However, many VQ applications not only require large-scale classifier systems $(M \geq 1000)$, but, in contrast to many ANN applications, *also* require systems which operate at high-speed ($\geq$ 1MHz) and which are capable of distinguishing between inputs which are very closely spaced (the order of 1μA in current-mode or 10mV in voltage-mode) [116], [117].

In this chapter, an integrated CMOS WTA network based on current-mode circuit principles with order of complexity $O(M)$ is described [19]. This system is highly

scalable and achieves higher *resolution*[1] than other current-mode WTA's, especially for large values of M. The operating speed of the WTA is independent of the presented input pattern and the system can therefore operate asynchronously, making best use of the underlying speed of the technology. In addition, since the overall speed and accuracy of the system degrade at a rate proportional to $\log_2 M$, which is a slowly varying function of M, especially for large M, such a WTA is especially suited to large-scale applications (e.g. VQ) where high speed and high resolution are simultaneous requirements. The circuit requires no additional DC current biasing and hence its power consumption is less than other current-mode WTA's [115]. Also, the identity of the winning input appears in the form of a binary word of dimension $\log_2 M$ without the need for any additional encoding circuitry, i.e., in the format required for VQ implementation.

The chapter begins with an introduction into VQ since the realisation of large-scale vector quantisers was the principal motivation for the development of the work presented in this chapter. In Section 5.2 the most popular architectures for WTA networks are described and compared based on the requirement for a WTA network capable of discriminating between a large number of inputs. Circuit realisation is dealt with in Section 5.3 followed by simulated and measured results for a 2-input WTA circuit and an 8-input WTA system in Section 5.4. The WTA's were fabricated using the MIETEC 2.4μm CMOS process. Finally, a WTA circuit for use in MFD's is presented.

5.1 Vector Quantisation

Data compression [120] is essential for reducing the data transmission requirements (bandwidth) and storage costs for applications such as digital broadcast TV/HDTV, videoconferencing, facsimile transmission of printed material, remote sensing, computer communication, and image database management. The fundamental goal of data compression is to reduce the bit rate for transmission or storage while preserving the essential fidelity of the source data. Various data compression techniques are available such as pulse code modulation (PCM), transform coding, predictive coding, and VQ. With the exception of VQ [18], all the other techniques have the common deficiency that quantisation or conversion from continuous quantities to discrete quantities is performed on *scalars* (e.g., on individual real-valued samples of analogue waveforms or pixels of

[1] Resolution here is defined as the minimum difference between the largest input variable and its nearest neighbour (as a function of the total dynamic range) for correct classification to occur.

images) and, thus, sub-optimal performance is achieved. This follows Shannon's source coding theorem, which states that asymptotic optimal performance is always achievable in theory by coding *vectors* (blocks) instead of scalars, even if the information source is memoryless [22].

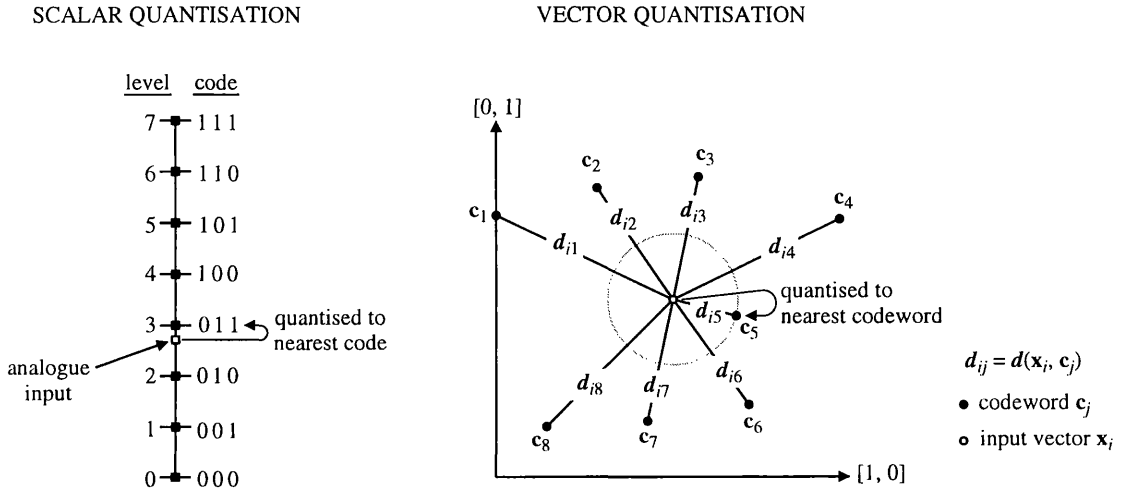


Figure 5.1 The difference between scalar and vector quantisation.

Figure 5.1 depicts the difference between scalar quantisation and VQ. With VQ fewer bits are required to encode a vector of samples compared to separate quantisation of individual samples and thus, higher levels of data compression are achieved. The penalty paid for the reduction in bit rate with VQ is an increase in the encoding implementation complexity compared to that required by scalar quantisation techniques. This is one of the primary reasons that prevented VQ from developing into a practical technique from a theoretical possibility. However, with the advances in very-large-scale-integrated (VLSI) technologies VQ has become a practical technique for speech and image compression at medium to low bit rates, and several analogue and digital vector quantisers have been developed in recent years [116]-[119], [121]. Other applications of VQ include pattern recognition and various ANN[2] architectures [122]. In addition, VQ is attractive for use in wireless communication systems where portable receivers are required because the decoding implementation complexity of VQ is much less than that of other data compression techniques for similar levels of performance. Finally, it should be noted that in the literature vector quantisers have also been called "vector A/D converters" [117].

[2] These are massively parallel-interconnected networks that emulate certain of the powerful organising principles found in the neurobiological systems.

5.1.1 General Overview

Figure 5.2 shows the block diagram of a VQ coding system where the data source is an image composed of pixels. In VQ, an m-dimensional vector $\mathbf{x}_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(m)})$, which represents a pixel when encoded, is represented as one of a finite set of M symbols. Associated with each symbol is an m-dimensional vector $\mathbf{c}_i = (c_i^{(1)}, c_i^{(2)}, \ldots, c_i^{(m)})$, called a *codeword*, and the complete set of M codewords is the *codebook* $C = \{\mathbf{c}_j,\ j = 1, 2, \ldots, M\}$. The codebook C is usually obtained through a training process using a large set of training data that is typical of the data that will be encountered in practice.

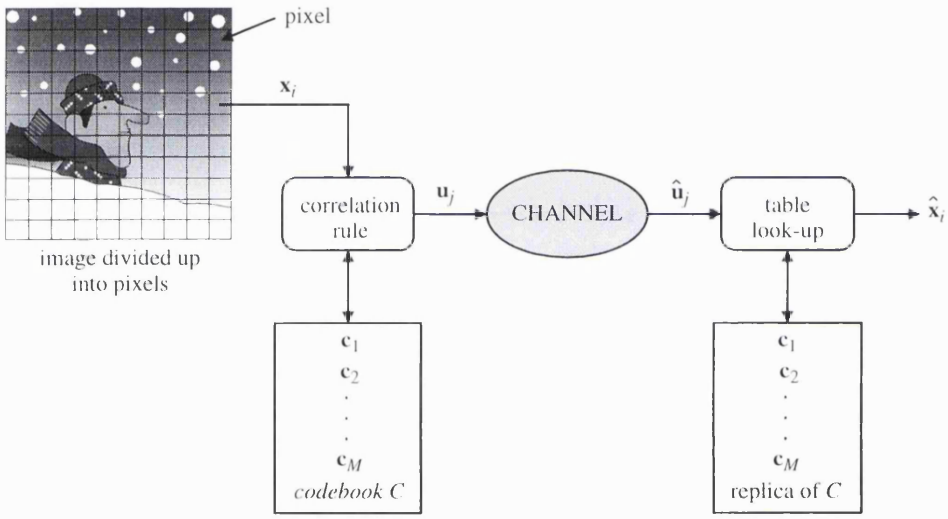


Figure 5.2 Block diagram of a VQ coding system.

During the encoding process, an input vector $\mathbf{x}_i$ that is to be encoded is compared to each of the M codewords in the codebook, and the *distortion* (or distance) $d(\mathbf{x}_i, \mathbf{c}_j)$ between the input vector and the codeword $\mathbf{c}_i$ is computed. The input vector $\mathbf{x}_i$ is then computed as the index $\mathbf{u}_j$ of the codeword that yields the minimum distortion. Usually, the index (or address) $\mathbf{u}_j$ appears in the form of a binary word of dimension $\log_2 M$, that is, $\mathbf{u}_j = (u_j^{(1)}, u_j^{(2)}, \ldots, u_j^{(\log_2 M)})$. The receiver has a copy of the codebook and uses this index to generate the reproduction codeword $\hat{\mathbf{c}}_i$ (i.e., the estimated input vector $\hat{\mathbf{x}}_i$). The quantity $R = \log_2 M$ is called the rate of the VQ in bits per vector, and $r = R/m$ is the rate in bits per symbol, or when the input is a sampled waveform, bits per sample [18]. Typically, $m = 16$, M ranges from 64 to 1024, an image is composed of 512×512 pixels, and the processing speed is about 6-10 Mvectors/s [120], [121].

One problem with VQ is that a large effort is required by the encoder to search the whole codebook in order to identify the nearest matching codeword to an input vector. Thus, instead of the *full-search* technique the *tree-search* technique may be employed yielding some loss in encoding performance. A tree-searched VQ encoder searches a sequence of small codebooks instead of one large codebook. The encoder structure can be depicted as a tree like a convolutional encoder (see Section 2.3.2), and each search and decision corresponds to advancing one level into the tree starting from the root of the tree. The drawback of tree-searched VQ encoders is that the codebook storage requirement is greater than that of the full search technique, and the codewords selected are not in general optimum in the sense of minimising $d(\mathbf{x}_i, \mathbf{c}_j)$ [123]. Other VQ techniques that reduce memory and encoding complexity can be found in [18].

5.1.2 Encoder Architecture

The block diagram of a full-search VQ encoder is shown in Fig. 5.3. The encoder consists of four sections namely, the buffering unit (BU), the codebook unit (CBU), the distortion computation unit (DCU), and the WTA network. The BU stores the input vectors one at a time, the CBU stores the M codewords, the DCU computes the distances between the input vector and each of the M codewords, and the WTA network selects the codeword closest to the input vector and encodes the winning codeword index. Additionally, the encoder may contain circuitry for updating the codebook contents (e.g., in an adaptive implementation) or in the case of an analogue realisation to also enable periodic refreshing of the codewords because of capacitor leakage effects [118].

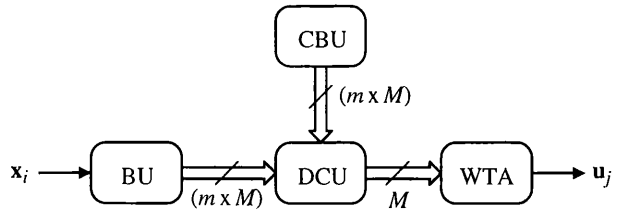


Figure 5.3 Simplified block diagram of a full-search vector quantiser.

The core of a full-search VQ encoder is the DCU, which consists of a $m \times M$ 2-D array of distance estimation cells. Each cell computes the distortion between one component of the input vector $\mathbf{x}_i$ and the corresponding component of one of the codewords $\mathbf{c}_j$. The

two most common distortion measures used in practice are the *mean-square error* (MSE) and the *mean-absolute error* (MAE) [123], given by

$$d(\mathbf{x}_i, \mathbf{c}_j) = \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2 = \frac{1}{m} \sum_{l=1}^{m} (x_i^{(l)} - c_j^{(l)})^2 \qquad 1 \le j \le M \tag{5-1}$$

and

$$d(\mathbf{x}_i, \mathbf{c}_j) = \left| \mathbf{x}_i - \mathbf{c}_j \right| = \frac{1}{m} \sum_{l=1}^{m} \left| x_i^{(l)} - c_j^{(l)} \right| \qquad 1 \le j \le M \tag{5-2}$$

respectively. The required accuracy of the computations is about 8-bits [116]. The resulting M distortion measures $d(\mathbf{x}_i, \mathbf{c}_j)$ are then presented to the WTA network, which selects the single winner

$$k^{\text{WTA}} = \arg \min_j d(\mathbf{x}_i, \mathbf{c}_j) \tag{5-3}$$

and encodes the winning codeword index $\mathbf{u}_j$. Note that the *min* function can be converted to a *max* function with the inversion of the sign of the distortion measures. Thus, a WTA network instead of a loser-take-all (LTA) network [124] can be employed.

5.2 WTA Architectures

WTA networks constitute a significant class of circuits in analogue and mixed-signal VLSI systems and are essential components of many systems such as ANN's, analogue vector quantisers and fuzzy logic systems. In fuzzy logic systems the WTA function is known as the MAX operator. Several integrated WTA designs that are compact and economical in terms of power dissipation have appeared in the literature [115], [116] [124]-[127]. These designs combine analogue circuit design techniques together with high-precision modern CMOS or BiCMOS technologies.

Depending on number of interconnects WTA networks can be classified as complexity $O(M)$ or complexity $O(M^2)$. WTA networks of complexity $O(M)$ are very attractive by virtue of the smaller silicon area required for their implementation, especially as M becomes large. The most frequently used approach to the implementation of WTA networks with order of complexity $O(M)$ is the *current-conveyor* [98]. This approach allows the design of WTA networks that can operate with current [115], [116] or voltage inputs [72]. In outline, the current-conveyor approach is very advantageous in terms of

transistor count but it relies on accurate transistor matching to preserve its accuracy, which can be difficult for large M.

A WTA network of complexity $O(M)$ can also be constructed using the *tree-structure* approach [117], [126]. This approach is highly modular and allows the design of large WTA networks since here accuracy is lost at a rate proportional to $\log_2 M$, which is a slowly varying function of M, compared to say current-conveyor WTA designs where resolution degrades at a rate proportional to M. The tree-structure approach is the one employed for the realisation of the integrated CMOS WTA network presented in this chapter. Other techniques for designing WTA networks of complexity $O(M)$ have been suggested. These are dealt with in [127].

WTA networks of complexity $O(M^2)$ are less attractive in terms of die area, especially as M becomes large. Nevertheless, there is a class of WTA networks of complexity $O(M^2)$ [124], [125] that require very small silicon area for their implementation. These designs are based on the *charge-sensing* approach and are built with minimum-geometry transistors but are slow.

The current-conveyor, charge-sensing and tree WTA networks are discussed in more detail in the subsections that follow. The advantages and disadvantages of each approach are described based on the requirement for a high-speed large WTA network, primarily for use in large-scale analogue VQ implementations.

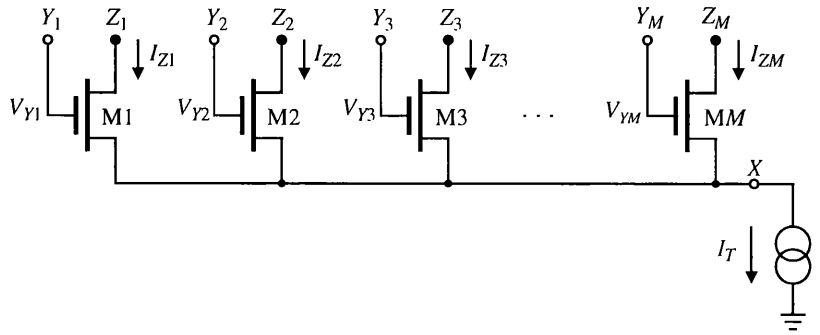


Figure 5.4 WTA network with one-transistor current conveyors.

5.2.1 Current-Conveyor WTA Networks

The current-conveyor approach allows the design of high-speed compact WTA networks to be constructed that can operate with voltage inputs or current inputs. A voltage-input

WTA network can be implemented using one-transistor current-conveyors [72] interacting with one another as shown in Fig. 5.4. Here, node X follows the largest input voltage V_{Yi}, turning off all other current-conveyors. The tail current I_T is then entirely conveyed (only if V_{Yi} is sufficiently larger from all others) to the output node Z_i identifying the ith input voltage as the largest. Note that the two-transistor version of this circuit is the well-known differential pair.

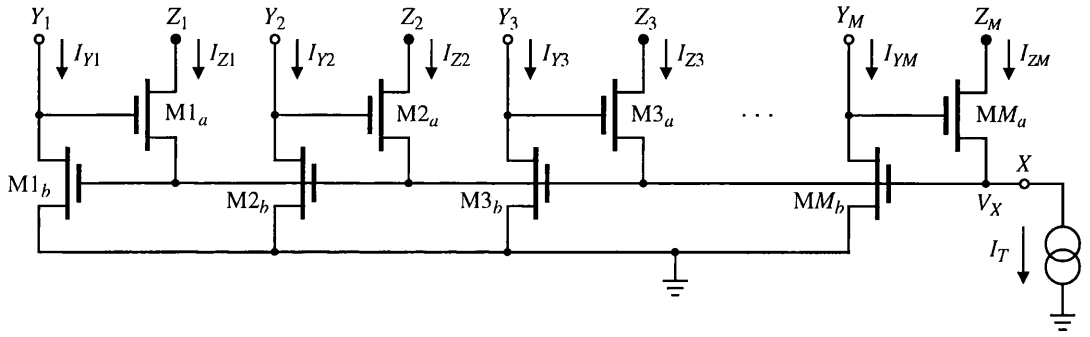


Figure 5.5 WTA network with two-transistor current-conveyors.

Two-transistor current-controlled current-conveyors can also interact with each other through a common communication line to implement a WTA network as shown in Fig. 5.5. This circuit was first suggested in [128] to operate in weak-inversion mode and was later presented in [115], operating in strong-inversion. Here, M two-transistor current-conveyors compete for the current supplied to the common communication line. This current, I_T, is steered to the output node Z_i of the conveyor with the largest input current I_{Yi}; and all other conveyors have zero output (again only if I_{Yi} is large enough). However, in its simplest form, such a system is limited in terms of M, since increasing M degrades the resolution of the WTA. This degradation is caused by:

1. the requirement that as M increases, more current must be conveyed to the winning conveyor, and

2. transistor matching degrades as a function of M.

Effect 1 can be compensated for by reducing the conveyor bias current, but at the expense of reducing the operating speed of the WTA [129]. Alternatively, both the resolution and speed of the current-conveyor WTA networks can be improved by the addition of positive feedback [129], but the transistor matching problem remains. The latter problem

is particularly significant when short gate length transistors are used, for example, in high-speed applications [115].

A general difficulty arises with current-conveyor WTA networks when accurate reproduction of the winning input is required to appear at the output of the WTA. In this case extra circuitry is required which increases the overall circuit size and power dissipation. Finally, in applications such as VQ implementation where the address of the winning codeword is required, a current-conveyor WTA network would require extra digital encoding circuitry [116], which is quite complex, especially when M is large. These problems are also encountered with the charge-sensing approach described next.

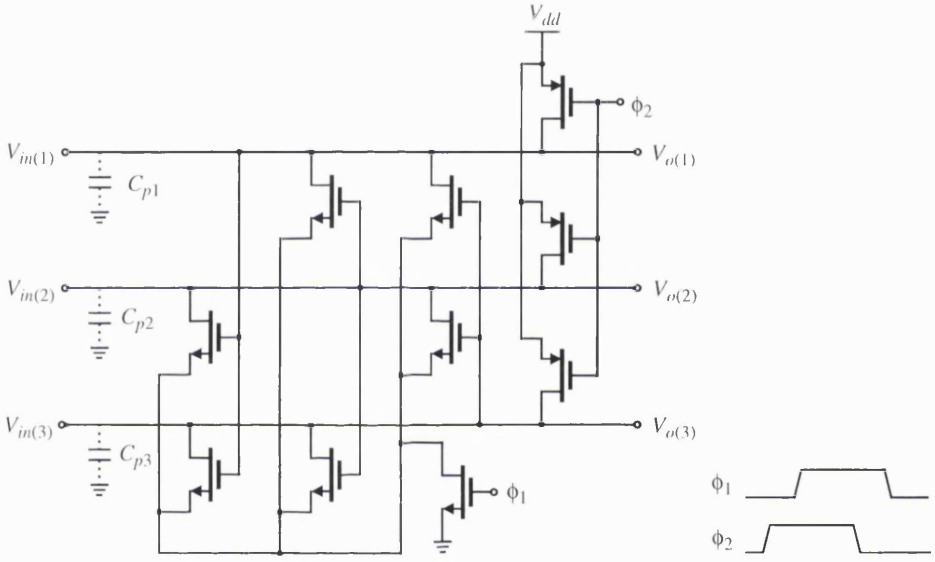


Figure 5.6 A 3-input charge-sensing WTA network.

5.2.2 Charge-Sensing WTA Networks

An alternative form of WTA networks employing the charge-sensing approach commonly used in DRAM's (dynamic random access memory) was proposed in [125]. A 3-input charge-sensing WTA network is shown in Fig. 5.6 where C_{p1}, C_{p2} and C_{p3} are usually equal parasitic capacitors The operation of the circuit is explained as follows. The input voltages $V_{in(1)}$, $V_{in(2)}$ and $V_{in(3)}$ are initially stored on the parasitic capacitors, and these voltages serve as the initial gate voltages of the NMOS transistors. The classification process is activated when ϕ_1 is pulsed high (see Fig. 5.6). This initiates the discharging of the capacitors through the NMOS transistors, and as a result the row voltages begin to decrease. Because the NMOS transistors are cross-coupled, the

decreasing of one row voltage forces some NMOS transistors to stop conducting. This process will continue until all but one row voltage remains above the threshold voltage V_{TH} of the NMOS transistors (all other row voltages continue to decrease until they are forced to ground). The row voltage, which remains above V_{TH} is the winner. At this point ϕ_2 is pulsed low to turn the PMOS transistors on and to further amplify the row voltages. This transition causes all row voltages to start an upward swing but only the row voltage with the initial advantage of at least one V_{TH} reaches V_{dd} (because of the cross-coupled NMOS interactions). All other row voltages are kept below V_{TH}.

The charge-sensing approach allows the design of WTA networks, which require very small silicon area for their realisation because minimum-geometry transistors are usually employed and the input load capacitors are parasitic capacitance. In addition no additional DC biasing is required and hence power dissipation is kept to a minimum. However, these designs are very limited in terms of M because of interconnect limitations. In addition, they are generally slower than the current-conveyor based types and can become unstable when presented with two or more equal inputs [125]. In general, their achievable resolution is limited by the offset from transistors (mainly variations in V_{TH}) and parasitic capacitance, and degrades at a rate proportional to $\sqrt{2}M$.

5.2.3 Tree-Structure WTA Networks

As an alternative to the current-conveyor and charge-sensing approaches, the tree WTA has been suggested as a possible WTA network with complexity $O(M)$ and both synchronous [117] and asynchronous [126] versions have been proposed. The asynchronous structure naturally lends itself to high-speed operation because it completely avoids the internal row-by-row clocking required in its synchronous counterpart. In addition, when the asynchronous tree WTA is realised using current-mode analogue circuits, which are renowned for their superior bandwidth and dynamic range properties, best use of the underlying speed of the chosen technology is made. In addition, many practical applications require a WTA with current inputs [116]. It should be noted that the asynchronous tree-structure approach is also employed in fuzzy logic systems for the realisation of the MAX and MIN functions [130].

The architecture of an asynchronous M-input current-mode tree WTA is shown in Fig. 5.7 and consists of $(M-1)$, 2-WTA current-mode cells. This network has $\log_2 M$ layers, each layer having half the number of 2-WTA's of that immediately above it. The block

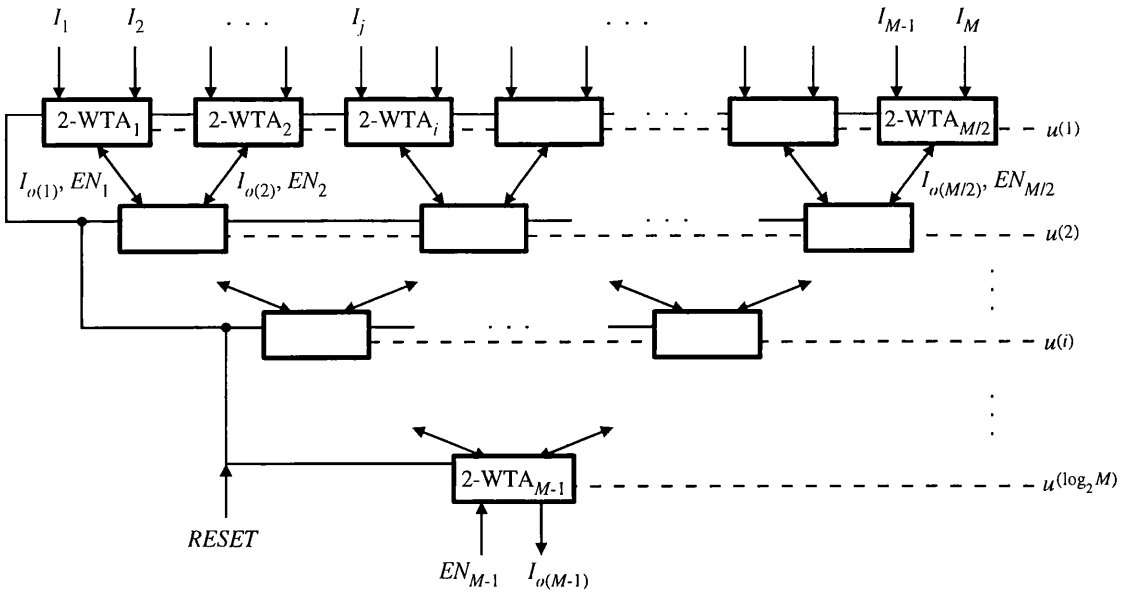


Figure 5.7 M-input tree WTA network.

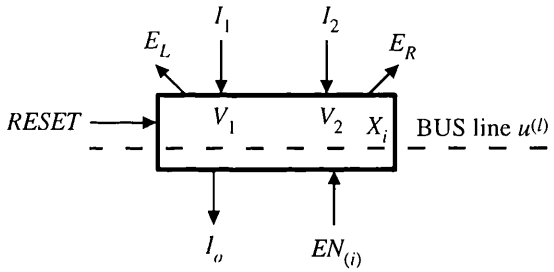


Figure 5.8 2-WTA cell.

Table 5.1 Relationships between the terminal variables of a 2-WTA cell.

Inputs	I_o	V_1	V_2
$I_1 > I_2$	I_1	1	0
$I_1 < I_2$	I_2	0	1

(a)

V_1	V_2	EN_i	E_L	E_R	X_i
1	0	0	0	0	O/C
0	1	0	0	0	O/C
1	0	1	1	0	0
0	1	1	0	1	1

(b)

schematic of a 2-WTA cell is shown in Fig. 5.8 and the relationships between the terminal variables are given in Tables 5.1a and 5.1b. The selection and identification of the largest (winning) current is carried out in two phases: a *forward pass* followed by a *backward pass* through the network. Each 2-WTA identifies the larger of its two input currents, I_1 and I_2, and produces an output current, I_o, which is a replica of this local 'winner' and is fed forward into succeeding rows of the tree. The output from the final 2-

WTA is thus a copy of the largest input (i.e., the global 'winner') current. This replication is a useful feature in many applications [116] and as discussed in Section 5.2.1, it is not available in current conveyor WTA networks without the addition of extra circuitry.

Although the accuracy of a single 2-WTA cell is excellent due to the local matching of its transistors, in a large tree network, especially for certain input patterns, errors accumulate due to the need to replicate currents many times over. An examination of Fig. 5.7 shows that the worst case occurs when I_1 and I_M are the largest and next largest inputs respectively, requiring that $\log_2 M$ replications of the competing currents be made before the final outcome can be decided. However, it should be noted that even in this worst case, resolution is lost at a rate proportional to $\log_2 M$ which, as already stated, is a slowly varying function of M, especially for large M. This should be compared to current-conveyor based WTA networks where resolution is lost at a rate proportional to M [127].

After the completion of the forward pass, the states of the 2-WTA's (i.e., voltages V_1 and V_2 in Table 5.1a) enable the winner to be identified determined and its point of origin to be represented by an encoded binary vector $\mathbf{u} = (u^{(1)}, u^{(2)}, \ldots, u^{(\log_2 M)})$, i.e., in the format required for VQ. The backward pass phase is initiated when the input enable voltage EN_{M-1} in the last 2-WTA cell of the tree goes from logical 0 to 1. Once enabled, the cell uses the voltages V_1 and V_2 to set the tri-state BUS X_{M-1} as shown in Table 5.1b. If the current came from the left (i.e., $V_1 = 1$) then E_R is set to 0 and hence, all the input currents numbered by $j > M/2$ are eliminated. The BUS line $u^{(\log_2 M)}$ (i.e., the most significant digit of the encoded word) is then set to 0 while the E_L of that 2-WTA cell is set to 1 and is used to activate the cell in the previous layer that steered in the winning current. Once activated, the above process is repeated in this new cell and the next significant bit in the encoded word is set. If the enable-input voltage EN_i of a cell is set to 0, the tri-state BUS X_i of that cell is an open-circuit (O/C), and hence cells within a layer that have not been enabled do not effect the output of the BUS line $u^{(l)}$ (where $1 \le l \le \log_2 M$) within that layer.

The classification process terminates when a 2-WTA cell in the first layer has been enabled. This means that all digits of the encoded word have been set and the identification number can be simply read, thus completing the classification process.

5.3 Circuit Realisation

Although the tree WTA described in [126] operates asynchronously, due to the characteristics of the 2-WTA cell employed in this design (see Fig. 5.9), the overall propagation delay of the *forward pass* is pattern dependent. The form of the pattern dependency in this case results in the propagation delay of each 2-WTA, depending not only on the magnitude of the largest of the two input currents (I_1, I_2) to the 2-WTA, but also on their relative difference $|I_1 - I_2|$. Hence a worst case guard period must be added to the actual propagation delay to ensure that the correct decision is made and, as a result, full advantage of the speed of the technology is not taken. These problems are addressed and eliminated in the CMOS 2-WTA circuit described next.

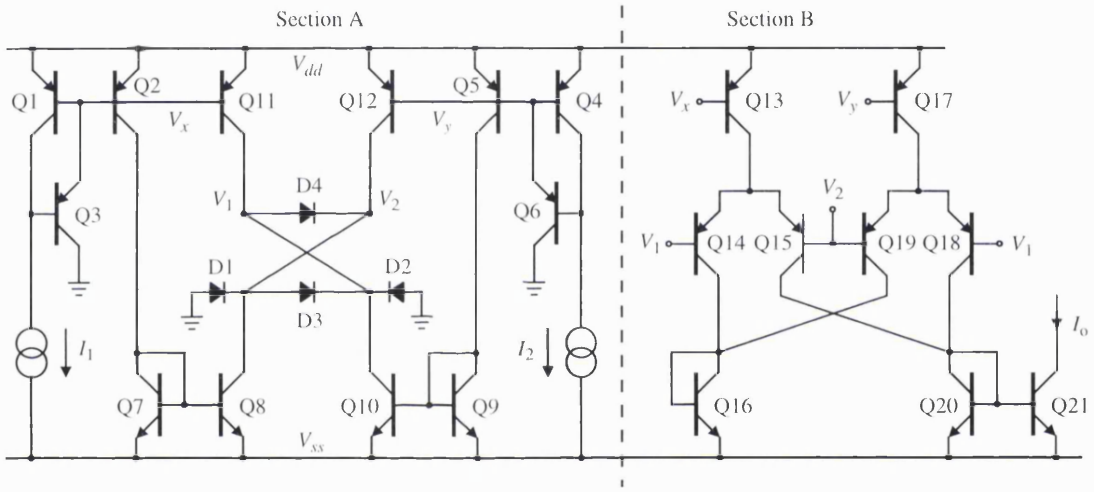


Figure 5.9 The forward pass circuitry employed in [126].

5.3.1 Forward Pass Circuitry

The forward pass is implemented by a circuit derived from that shown in Fig. 5.10, which was originally proposed for use in fuzzy logic systems [130]. Its function is to identify the larger of the two input currents I_1 and I_2 and to produce a copy of this local 'winner' at the output, I_o. In Fig. 5.10, the competing input currents are I_1 and I_2 and these initially flow through the diode-connected transistors M1 and M3, turning on the mirror transistors M2 and M4. As the diode voltages V_1 and V_2 increase, the cross-coupled latch transistors M5 and M6 begin to conduct and hence reduce the input currents to the diodes competitively until the smaller current is forced to zero. The winning current is mirrored to the output by means of either M2 or M4 and the output PMOS mirror (M7, M8).

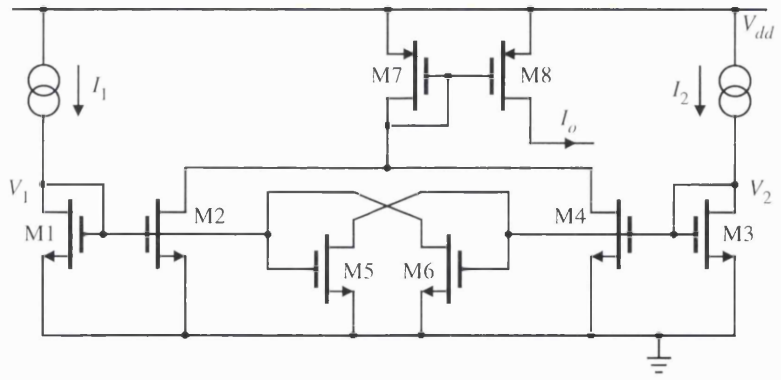
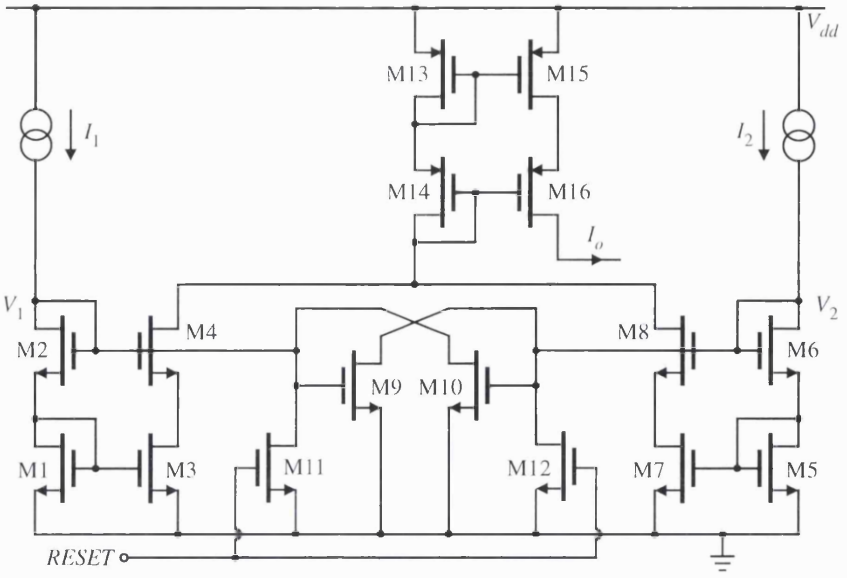


Figure 5.10 A possible forward pass circuitry.

However, this simple arrangement has several disadvantages. In particular, the current drive capability of the latch transistors M5 and M6 is limited, adversely affecting both the accuracy and the operating speed of the 2-WTA. The gate widths of M5 and M6 must be at least the same as M1 and M3 to allow convergence to occur when I_1 and I_2 are very similar. On the other hand, if the width of the latch transistors is increased, the resulting capacitive loading more than outweighs the increased current drive and actually reduces the operating speed. In addition, the circuit in Fig 5.10 is very susceptible to the effects of channel length modulation, which further reduces the accuracy of the replicated output current, I_o. Both these problems are much reduced in the improved CMOS 2-WTA shown in Fig. 5.11.

In the circuit in Fig. 5.11, the input currents I_1 and I_2 are applied to a pair of NMOS cascode current mirrors (M1-M4 and M5-M8, respectively) and the diode voltages V_1 and V_2 operate the latch transistors M9 and M10, as in the circuit in Fig. 5.10. Unlike the circuit in Fig. 5.10, however, there are two diode drops available to turn on M9 and M10 and hence the current drive capability of the latch is much enhanced. Consequently, since M9 and M10 are turned on *before* the diodes begin to conduct, the rise times of V_1 and V_2 are essentially fast linear ramps due to the charging of the relevant parasitic capacitors. The drains of M9 and M10 have the capability to sink several times the competing input currents, guaranteeing rapid convergence. The circuit thus achieves high-speed operation in addition to increased accuracy in the replication of the winning current due to the use of cascode mirrors. A full description of the operation of this CMOS 2-WTA circuit was given in Section 4.2.1. Finally, at the completion of the WTA process, the latches are reset globally by means of the transistors M11 and M12 and by



Transistor	M1 - M10	M11, M12	M13 - M16
W (μm)	20	4.8	40
L (μm)	2.4	2.4	2.4

Figure 5.11 Forward pass circuitry and transistor dimensions.

the *RESET* line. Note that the reset line should be high when the inputs are set up, pulsed low to start the competition and then set high again to reset the latches.

The operating speed of the CMOS 2-WTA circuit is essentially limited by the rise times of the three cascode current mirrors (M1-M4, M5-M8 and M13-M16) and these depend on the individual input currents I_1 and I_2 and *not* on the difference between them, thus avoiding the inconvenient pattern dependence reported in [126]. In addition, a tree WTA network constructed from the 2-WTA circuit in Fig. 5.11 requires less stages of current copying than other current-mode tree WTA designs [126], [130]. This is reflected in greater accuracy, compactness, operating speed and lower power dissipation for a given M than the competing designs.

Note that although the propagation delay of the circuit in Fig. 5.11 still depends on the magnitude of the winning input current, this is an advantage in the case of a tree WTA network. This form of pattern dependency ensures that the winning current will always complete its forward pass through the tree faster than any competing paths. In consequence, it is not necessary to reset the 2-WTA cells row-by-row as in [117], but only once per complete cycle. Since the WTA can be described as *internally*

asynchronous, only one *RESET* pulse and associated guard period is required per complete WTA cycle.

Notice also that although the select-max section of the 2-WTA circuit in Fig. 5.11 is based on NMOS transistors, if alternate rows of the tree are realised with PMOS transistors, it is possible to eliminate the output current mirror of each cell (M13-M16) in Fig. 5.11. This halves the number of current replications in each forward path, thus improving the resolution of the WTA for a particular value of M. This modification reduces the power consumption of the tree and the number of transistors employed. It also increases the overall operating speed of the system to some extent, although the speed increase is limited somewhat by the slowness of PMOS mirrors compared to their NMOS equivalents. Note that the PMOS latches would have to be driven by an inverted version of the *RESET* pulse. This only requires the addition of one INVERTER per complete system. The relation between the terminal variables of a 2-WTA cell in the PMOS rows of such a tree WTA would be according to Tables 5.2a and 5.2b.

Table 5.2 Relationships between the terminal variables of a 2-WTA cell for use in the PMOS rows of a tree WTA network.

Inputs	I_o	V_1	V_2
$I_1 > I_2$	I_1	0	1
$I_1 < I_2$	I_2	1	0

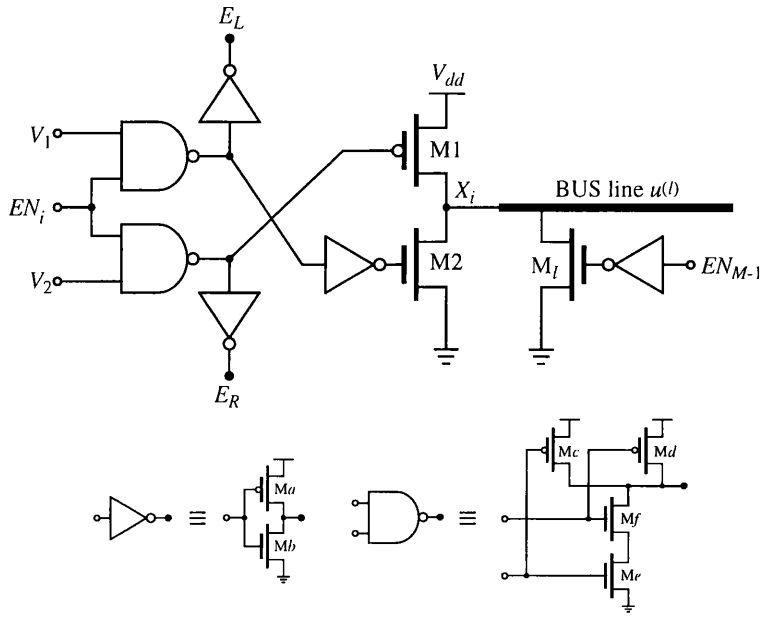
(a)

V_1	V_2	EN_i	E_L	E_R	X_i
1	0	0	0	0	O/C
0	1	0	0	0	O/C
1	0	1	0	1	1
0	1	1	1	0	0

(b)

5.3.2 Backward Pass Circuitry

At the completion of the forward pass, the value of the winning input current $I_{o(\max)}$ is known. Also, the voltages V_1 and V_2 are set high or low for each 2-WTA cell, corresponding to logic levels {1, 0}. As described in Section 5.2.3, the purpose of the backward pass is to identify the origin of $I_{o(\max)}$ and to represent the point of origin by a binary encoded word of dimension $\log_2 M$. The backward pass phase is initiated when the EN_{M-1} line goes from logical 0 to 1. Although this function is supplied externally for the purposes of the measurements presented in this chapter, in a real application, it can be derived very simply from the output of a 2-input XOR gate whose inputs are V_1 and V_2 of the last 2-WTA cell.



Transistor	M1	M2	M3	M*a*	M*b* - M*e*	M*f*
W (μm)	80	40	4.8	16	4.8	14.4
L (μm)	2.4	2.4	2.4	2.4	2.4	2.4

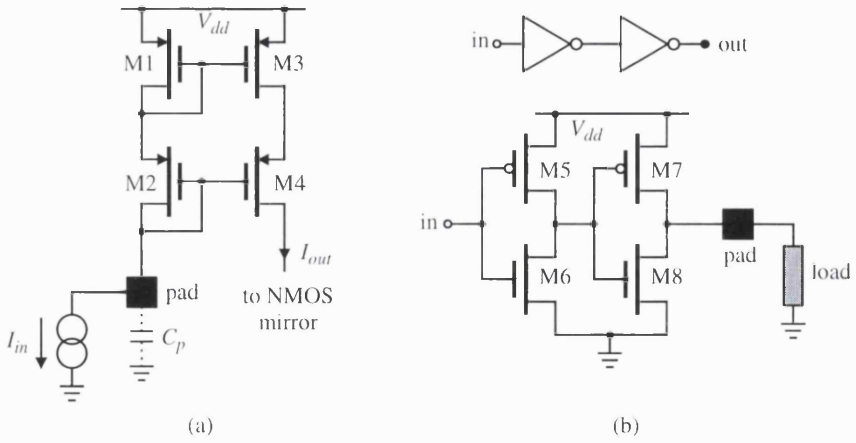
Figure 5.12 Backward pass circuitry and transistor dimensions.

The logic circuitry of each 2-WTA cell is shown schematically in Fig. 5.12. The NAND gates and INVERTERS are standard CMOS units with certain transistors in the NAND gates suitably scaled to accommodate non-standard logic levels (2 to 3V). The transistors M1 and M2 realise a tri-state buffer, which is connected to the BUS line $u^{(l)}$ of that tree row as shown in Fig. 5.12. An additional transistor, M_l, is required for each row of the tree WTA all connected to the EN_{M-1} line by means of a single buffer. This arrangement defines the state of the tri-state BUS at ground when EN_{M-1} is low, i.e., for the duration of the forward pass.

Finally, at the completion of the WTA process, the latches are reset globally by means of the transistors M11 and M12 when the *RESET* line goes high.

5.3.3 Input and Output Buffers

Because of the discharging action of the *RESET* pulse, additional input PMOS current mirror buffers were employed (see Fig 5.13a). These buffers were connected to the input NMOS current mirrors of a single CMOS 2-WTA circuit and, in the case of a tree WTA network, to the inputs of the first row of 2-WTA cells. These input buffers prevent



Transistor	M1 - M4	M5	M6	M7	M8
W (μm)	50	60	20	240	80
L (μm)	2.4	2.4	2.4	2.4	2.4

Figure 5.13 Buffers; (a) input, (b) output.

discharging of the large pad parasitic capacitance when the *RESET* line goes high, and thus allow easier testing and real-time measurements of the classification period.

To be able to drive the pads and external loads, output buffers were also necessary. These buffers (see Fig. 5.13b) are formed by cascoding two INVERTERS with transistors suitably scaled to drive loads of up to 15pF. These buffers were connected internally to all digital outputs of the fabricated prototype chips.

5.4 Simulated and Measured Results

A CMOS 2-WTA circuit prototype of the type described in Section 5.3 and an 8-input tree WTA have been fabricated using the MIETEC 2.4μm CMOS process with 5V supplies and with HSPICE parameters supplied by EUROCHIP [131]. The simulations described in the next sub-sections are based on extracted versions of the complete circuits and hence include all layout parasitics.

The experimental set up for the CMOS 2-WTA chip is shown in Fig. 5.14. To be able to measure the value of the winning current I_o, a 50Ω resistor was used to convert this current into a proportional output voltage, which was in turn displayed on an analogue oscilloscope (scope 1 in Fig. 5.14) using the (10×) multiplication function of this oscilloscope. The input current sources I_1 and I_2 in Fig. 5.14 were implemented using two LM334 variable current sources each connected in parallel with a variable resistor to

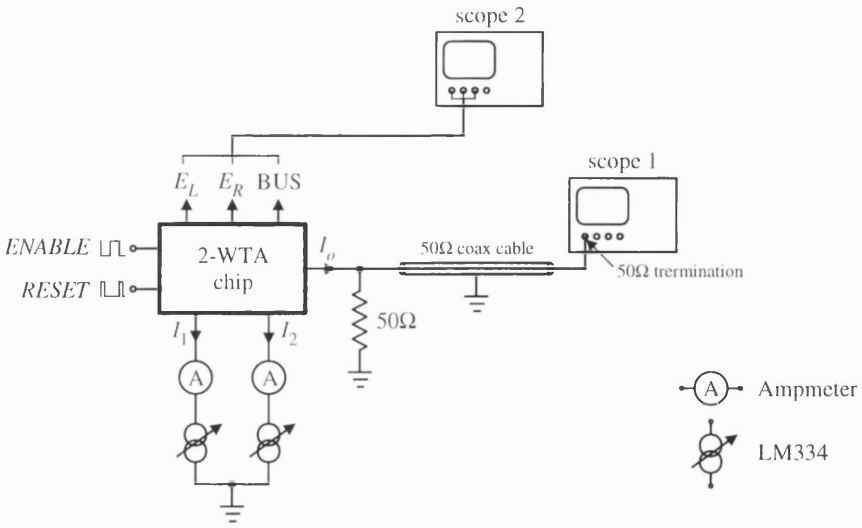


Figure 5.14 Experimental setup for testing the CMOS 2-WTA chip.

generate variable current signal levels (in the range or 10-120μA). The digital outputs from the chip were displayed on a similar oscilloscope (scope 2 in Fig. 5.14). A similar arrangement to the one shown in Fig. 5.14 was also used for testing the prototype 8-input CMOS WTA chip.

5.4.1 Forward Pass Circuitry Results

Two sets of simulations were carried out to determine the properties of the CMOS 2-WTA circuit in its forward pass mode. In both cases, the circuit was triggered by a negative-going step being applied to the *RESET* line. Firstly, I_1 was held constant at 5μA while I_2 was increased from 10μA to 100μA in steps of 2.5μA (the maximum current is limited to about 110μA in this example by the current mirrors leaving saturation). The *forward pass classification period*, τ_f, is defined as the period between the point where the *RESET* pulse falls to 50% of its initial value and I_o reaching 90% of its final value. The results of this simulation, together with experimental results are given in Fig. 5.15. A microphotograph of the CMOS 2-WTA chip is shown in Fig. 5.16 (note that this chip includes certain test structures such as output buffers, etc.).

In the second simulation, $(I_2 - I_1)$ was held constant at 5μA while I_2 was once again varied from 10μA to 100μA in steps of 2.5μA. These results are indistinguishable from the first set, i.e., τ_f depends only on the value of the largest input current, I_2 in this case, and not on $|I_1 - I_2|$. Further measurements indicated that the circuit could differentiate between currents differing by less than 1μA with negligible loss of operating speed. The

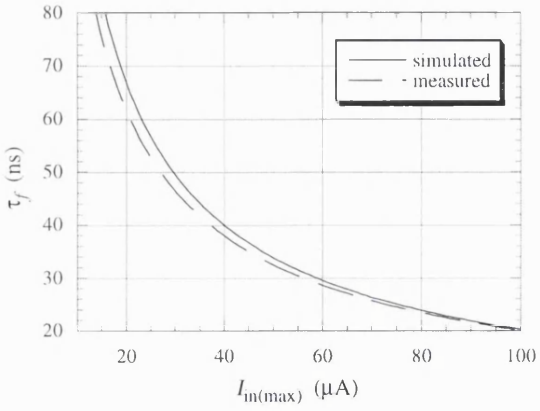


Figure 5.15 2-WTA performance curves.

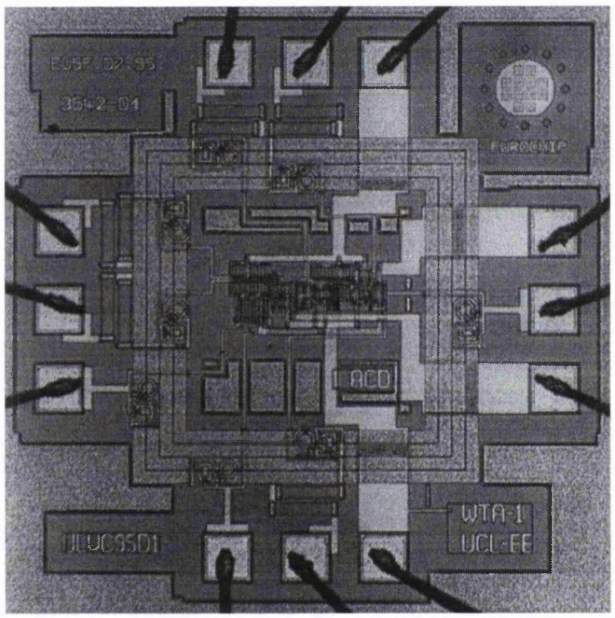


Figure 5.16 Microphotograph of the 2-WTA chip.

steady state-state power dissipation of a 2-WTA cell with 5V supplies and with $I_1 = 10\mu A$ and $I_2 = 15\mu A$ is 200μW. This should be compared with 450μW for the circuit described in [126] under similar operating conditions. For a tree WTA network with $\log_2 M$ rows the *total forward pass classification period*, T_f, is given by

$$T_f = \tau_{f(\max)} \cdot \log_2 M \tag{5-4}$$

where $\tau_{f(\max)}$ is the classification period corresponding to the largest (winning) input current $I_{in(\max)}$.

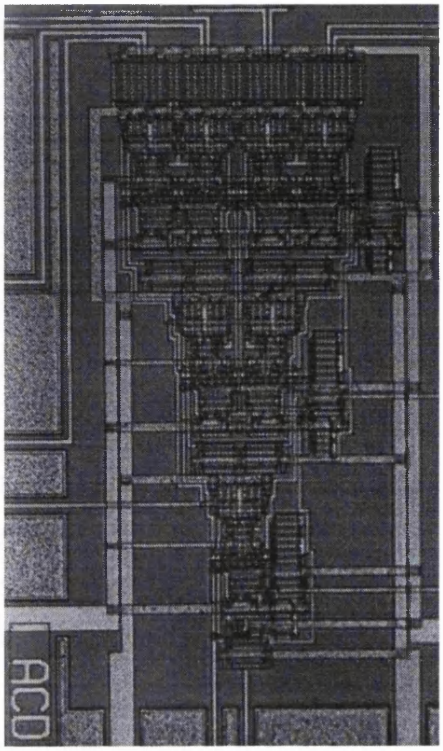


Figure 5.17 Microphotograph of the 8-input WTA network.

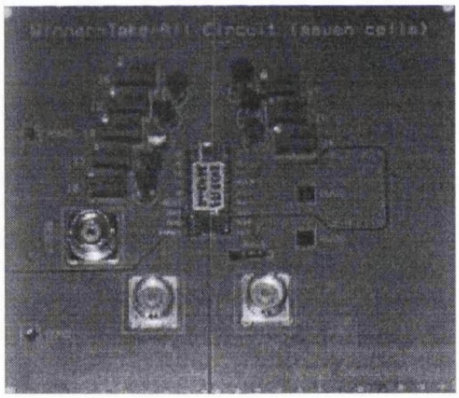


Figure 5.18 The PCB used for testing the 8-input WTA chip.

Measurements were carried out on a tree WTA network with $M = 8$, a microphotograph of which is shown in Fig. 5.17 (note that this chip also includes certain test structures such as output buffers, etc.) and a photograph of the PCB used for testing is shown in Fig. 5.18. The active area of each 2-WTA cell is $80\mu m \times 280\mu m$ and because of the tree-structure very little area is occupied by interconnect. Two input currents of $100\mu A$ and $99\mu A$ were applied and the remaining six input currents were set at arbitrarily chosen smaller values. The two largest inputs were initially placed at the positions I_1 and I_2 in Fig. 5.7 (i.e., best case matching) and in a subsequent experiment at I_1 and I_8 (i.e., worst

case matching). In both cases, classification was successful over many trials carried out on each of the 10 examples of the fabricated system. In all cases, T_f was 60ns, i.e., $3\times\tau_{f(\max)}$ for a single cell in Fig. 5.15. This WTA therefore achieves a measured resolution of better than 1%, which should be compared with 6% [115] and 1.5% [127] for other current mode WTA networks with the same winning input current.

5.4.2 Backward Pass Circuitry Results

HSPICE analysis of the arrangement shown in Fig. 5.12 gave a delay, τ_e, of 2ns from the setting of the EN_i line to the point where E_L or E_R reach 90% of V_{dd}. For the purposes of simulation, the tri-state BUS X_i was loaded with a capacitance of 2pF and the additional delay required to set the tri-state BUS, high (90% V_{dd}) or low (10% V_{dd}) was 3ns[3]. As mentioned in Section 5.3.2, at the completion of the WTA cycle, the *RESET* line goes high, resetting all the variables globally. Simulations indicate that this adds approximately 10ns to the total cycle time. Using Eq. (5-4), for a tree WTA network with $\log_2 M$ rows, the *total classification period*, T_s (expressed in ns), is given by

$$T_s = \{(\tau_e + \tau_{f(\max)})\cdot\log_2 M\}+13 \tag{5-5}$$

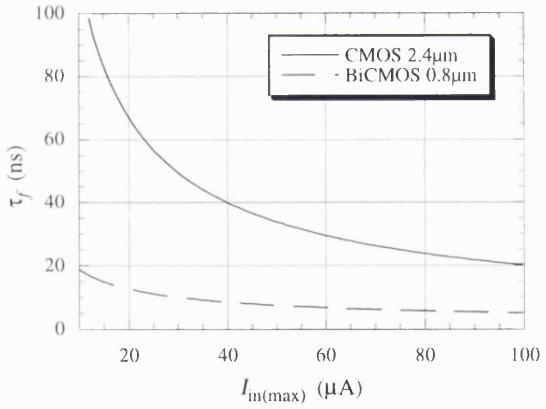


Figure 5.19 Comparison of the operating speed of the BiCMOS 2-WTA in Fig. 4.7 and the CMOS 2-WTA in Fig. 5.11.

As an example, consider the design of a tree WTA network with $M = 1024$ inputs, and where the maximum input current is 100μA. From Fig. 5.15, $\tau_{f(\max)} = 20$ns and so T_s is 233ns. Finally, in order to investigate the speed improvement and power dissipation

3 In measurements an extra 5ns delay was observed because of the output buffers and the larger capacitive loads.

reduction when the BiCMOS 2-WTA circuit in Fig. 4.7 is used to implement the forward pass circuitry, the simulations performed for the CMOS 2-WTA (see Fig. 5.15) were repeated for the BiCMOS circuit [132]. The results of this simulation are given in Fig. 5.19. In addition to enabling operation at 2.8V supply rails, the BiCMOS 2-WTA circuit provides enhanced operating speed. Thus, for the example considered above, T_s reduces to 78ns ($\tau_e = 2$ns and the *RESET* delay is 8ns).

5.5 A WTA Circuit for the MFDA

As mentioned in Section 3.7.3, the function of the WTA network in a practical MFD is to *only* establish whether the path with the largest metric (among 2^K survivors) lies in the upper or lower subset, corresponding to a "1" or a "0" decoding decision. An efficient current-mode WTA circuit for performing this task is shown in Fig. 5.20. The circuit is divided into two sections as indicated by the dashed line. The left hand-side section accepts the survivor path metric currents from the upper modified tree and the other section the survivor path metric currents from the lower modified tree. The core of the circuit is the current-conveyor consisting of $Q_1 - Q_{2^K}$ (its operation was described in Section 5.2.1) and whose outputs are connected into two groups as shown in the circuit. The two output currents $I_{o(\text{up})}$ and $I_{o(\text{lo})}$ carry the information required to establish the position of the survivor with the largest metric. For example, if this survivor originated in the upper modified tree, $I_{o(\text{up})}$ would be equal to the tail current I_T, while $I_{o(\text{lo})}$ would be zero. These currents can be easily converted into a "1" or a "0" decoding decision by a trivial logic circuit.

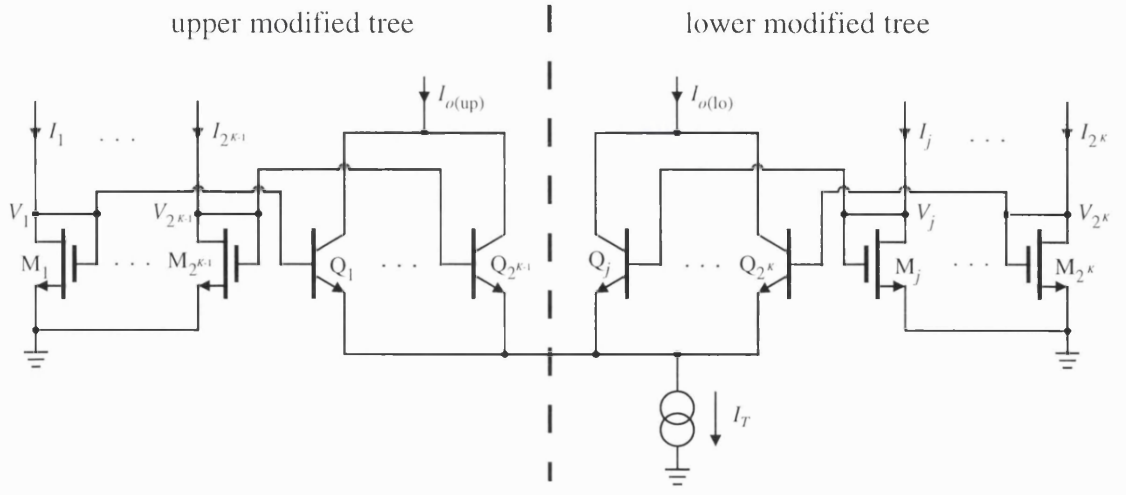


Figure 5.20 A possible circuit realisation of the WTA network of a MFD.

Chapter

6

Conclusions and Future Work

AS the millennium approaches digital communications are finding applications in new areas such as digital television, digital audio broadcasting, satellite telephony and advanced wireless systems. These applications demand high operating speeds, smaller transceiver size, lower power dissipation and low-voltage operation. To date mainly digital circuit techniques have been used in an attempt to meet these requirements. This has led to the dominant position of digital circuits in the implementation of the baseband section of digital communication systems, while analogue circuits are used mainly in the data conversion subsystem and the radio frequency/intermediate frequency (RF/IF) sections.

One of the aims of this thesis was to increase awareness of the potential for using analogue circuit techniques in signal processing subsystems to improve performance in these key areas of speed, size, voltage and power dissipation. In particular, the *current-mode* analogue approach has been applied to the design of convolutional decoders one of which employs the VA in a standard hybrid analogue/digital architecture and the other employing a new architecture, the MFDA. This latter architecture was developed through this thesis to be particularly suitable for analogue implementation. The current-mode approach has also been applied to the implementation of vector quantisers for data compression because of the similarities in the architectures of these systems.

The remainder of this chapter contains a brief summary of the contents of the previous chapters of this thesis excluding the introduction in Chapter 1. This is followed by a description of future planned work.

6.1 Summary and Conclusions

In Chapter 2 an overview of coding for error-control in digital communication systems was presented. The use of error-control coding provides a more economical solution to improving transmission performance than through the use of other methods such as higher power transmitters and larger antennae. The emphasis here was placed on FEC using convolutional codes. The large values of CG and the ease with which soft convolutional decoders can be implemented has led to the widespread use of theses codes in a variety of digital communication systems. The fundamentals of convolutional codes were presented, and key applications were discussed at the end of the chapter.

Convolutional decoding techniques were covered in Chapter 3. The early decoding techniques, namely sequential and feedback decoding, are sub-optimum because they perform a sparse search of the trellis paths (for reasons of implementation complexity), and generally do not employ the ML metric. The VA, however, provides a practical method for realising MLSD for applications such as convolutional codes and digital magnetic recording systems. Within the convolutional decoding framework, a new decoding algorithm/architecture, the MFDA was developed. This algorithm embodies features of both the VA and the feedback decoding technique, and can be viewed as equivalent to a truncated metric and path memory realisation of the VA. For the specific codes considered, the coding performance of the MFDA was shown by simulations to closely approach that of the truncated VA. However, unlike a conventional Viterbi decoder, the MFDA enables a soft convolutional decoder to be constructed *almost entirely* from analogue components and subsystems because it completely eliminates the need for any (digital) SSM block. The MFDA makes possible a much better flexible trade-off between speed, size and power dissipation than is possible with the hybrid analogue/digital VA approach. This suggests a highly novel and advantageous approach to the design of convolutional decoders for use in applications where size and power dissipation rather than ultimate speed are crucial parameters (e.g., certain types of future wireless systems).

To date, *voltage-mode* analogue subsystems have been employed successfully to improve the performance of class-IV Viterbi detectors for disk drive applications, and Viterbi decoders for more general applications (e.g., convolutional decoding), in terms of speed, complexity and power dissipation. The principal aim in Chapter 4 was to investigate the use of *current-mode* analogue subsystems for the implementation of general classes of analogue Viterbi decoder. Current-mode circuits offer many potential advantages over their voltage-mode equivalents: wide bandwidth and dynamic range and low-voltage operation being among the outstanding features of this approach. A range of current-mode CMOS/BiCMOS candidate ACSU's were designed in 0.8μm technologies which showed that this approach offers a more attractive trade-off to be obtained between speed, size and power consumption in addition to allowing the supply rails to be reduced from 5V to 2.8V. The most striking feature of these miniature ACSU's is their extremely low power dissipation (<1.7mW) which is an order of magnitude less than that of a recently reported voltage-mode ACSU for general Viterbi decoding applications. Simulations showed that the potential operating speed of a general class hybrid analogue/digital Viterbi decoder employing the proposed approach is about 50Mbit/s. Related issues such as maximisation of dynamic range, decoder front-end and distance computation circuitry were also dealt with in this chapter. The current-mode approach was also extended to the realisation of the MFDA where suitable circuits were presented.

A scalable high-speed asynchronous current-mode CMOS WTA circuit of complexity $O(M)$, where M is the number of inputs, was presented in Chapter 5. The new WTA has improved resolution and operating speed compared to other current-mode WTA's, especially for large M. The proposed arrangement is, therefore, well suited to applications requiring large WTA systems where operating speed and resolution are important parameters (e.g., VQ systems). Also, the identity of the winning input appears in the form of a binary word of dimension $\log_2 M$ without the need for any additional encoding circuitry (i.e., in the format required for VQ). Two versions of the proposed WTA were made using the MIETEC 2.4μm CMOS process with 5V nominal supply rails. The first version was a 2-WTA subsystem and the other was an 8-input WTA system made up of seven 2-WTA cells. Each of these cells has an active area of 0.025mm^2. Measurements showed that the proposed WTA circuit could resolve input currents differing by less than 1μA with negligible loss of operating speed. Detailed measured results and simulations were presented.

6.2 Future and Planned Work

As mentioned in Chapter 1 the work of this thesis has resulted in a three year grant from the U.K. Engineering and Physical Science Research Council (EPSRC) with effect from 1st April 1998. Within this time frame it is intended to build two sets of $R = 1/2$ analogue soft convolutional decoders using BiCMOS technology and to critically compare their performance with decoders reported in the current literature in terms of speed, coding performance, die size and power dissipation. The first decoder will employ the VA in a standard architecture using both analogue and digital subsystems, while the second will be based on the MFDA. The work will also be extended to the implementation of an EPR4 Viterbi detector for disk drive read/write IC's. These decoders will be largely based on the circuit ideas presented in Chapter 4. In the case of the MFDA, special attention will be given to the N-step trellis principle to establish its limits within this algorithm. This will include a simulation phase, which will take into account all possible analogue errors to establish their effects on the coding performance. Also, a less hardware intensive implementation of the SS block than the one proposed in Chapter 4 will be investigated.

A possible area of future work might include for example the implementation of *soft-output* Viterbi decoders in the analogue domain. A soft-output Viterbi decoder accepts and delivers soft sample values by estimating the reliability of the symbol decisions [133]. These types of decoder have recently been applied to many applications requiring some form of concatenation (e.g., concatenated convolutional coding and coded Viterbi equalisation) because they improve the S/N gain. The drawback of such a decoder is the fact that its implementation complexity is about twice that of a conventional Viterbi decoder. This is evident in the digital realisations reported so far. However, realising a Viterbi decoder with soft decisions is perhaps better suited to analogue implementation. This will make soft-output Viterbi decoders the preferred soft-output decoding technique for high-speed applications requiring small and power-efficient ASCI implementations.

The WTA system in Chapter 5 was classified as ideal for the implementation of large analogue vector quantisers. Future work may extend to implementing a complete analogue vector quantiser, with say 256 codewords, using the presented WTA approach. This would make a challenging and potentially useful project in its own right since it would require the design of a high-speed and area/power efficient distortion computation cell with an accuracy of about 8 equivalent bits. The idea might also be extended to

implementing an adaptive analogue vector quantiser. In this case an efficient technique for updating the codebook contents would be required.

Finally, there is much interest in the application of analogue circuits and systems in the design of low-voltage and low-power electronic systems. This is because in many cases, certain signal processing applications cannot be implemented digitally (in hardware or software) while meeting the required constraints of size, speed and power consumption. Since Viterbi decoding is employed in a variety of electronic systems, the opportunities for analogue designers to explore new territories for analogue Viterbi decoders are limitless.

References

[1] 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.

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

[3] Qualcomm Inc., Forward Error Correction Data Book, Feb. 1997.

[4] R. W. Wood and D. A. Petersen, "Viterbi detection of class IV partial response on a magnetic recording channel," *IEEE Trans. Commun.*, vol. COM-34, pp. 454-461, May 1986.

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

[6] R. G. Yamasaki *et al.*, "A 72Mb/s PRML disk-drive channel chip with an analogue sampled-data signal processor," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, Feb. 1994, pp. 278-279.

[7] K. Parsi *et al.*, "A PRML read/write channel IC using analog signal processing for 200Mb/s HDD," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1817-1830, Nov. 1996.

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

[9] K. Fukahori *et al.*, "An analog EPR4 Viterbi detector in read channel IC for magnetic hard disks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, Feb. 1998, pp. 380-381.

[10] K. D. Fisher *et al.*, "PRML detection boosts hard-disk drive capability," *IEEE Spectrum*, vol. 33, pp. 70-76, Nov. 1996.

[11] G. Cherubini, S. Olser, and G. Ungerbroeck, "A quaternary partial-response class IV transceiver for 125Mbit/s data transmission over unshielded twisted-pair cables: Principles of operation and VLSI realisation," *IEEE J. Select. Areas Commun.*," vol. 13 pp. 1684-1691, Dec 1995.

[12] 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.

[13] M. H. Shakiba, D. A. Johns, and K. W. Martin, "BiCMOS circuits for analogue Viterbi decoders," accepted for publication in *IEEE Trans. Circuits Syst.*, Dec. 1997.

[14] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS scaling for high performance and low power: The next ten years," *Proc. IEEE*, vol. 83, pp. 595-606, Apr. 1995.

[15] C. Toumazou, F. J. Lidgey and D. G. Haigh, Eds., *Analogue IC Design: The Current-Mode Approach.* London: Peter Peregrinus, 1990.

[16] C. Toumazou, J. B. Hughes and N. C. Battersby, Eds., *Switched-Currents: An Analogue Technique for Digital Technology.* London: Peter Peregrinus, 1993.

[17] Chih-Lin I *et al.*, "IS-95 Enhancements for multimedia services," *Bell Labs Tech. J.*, pp. 60-87, autumn 1996.

[18] R. M. Gray, "Vector quantization," *IEEE Acoust. Speech Signal Processing. Mag.*, vol. 1, pp. 4-29, Apr. 1984.

[19] 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.

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

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

[22] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, Jul. 1948, and vol. 27, pp. 623-656, Oct. 1948.

[23] A. Konheim, *Cryptography: A Primer.* NY: Wiley, 1981.

[24] J. G. Proakis, *Digital Communications.* NY: McGraw-Hill, 1995.

[25] B. Sklar, "Defining, designing, and evaluating digital communication systems," *IEEE Commun. Mag.*, vol. 33, pp. 92-101, Nov. 1993.

[26] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets" *IEEE Commun. Mag.*, vol. 25, pp. 5-21, Feb. 1987.

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

[28] V. K. Bhargava, "Forward error correction schemes for digital communications," *IEEE Commun. Mag.*, vol. 21, pp. 11-19, Jan. 1983.

[29] H. O. Burton and D. D. Sullivan, "Errors and error control," *Proc. IEEE*, vol. 60, pp. 1293-1301, Nov. 1972.

[30] I. M. Jacobs, "Practical applications of coding," *IEEE Trans. Info. Theory,* vol. IT-20, pp. 305-310, May 1974.

[31] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 10, pp. 300-304, Jun. 1960.

[32] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751-772, Oct. 1971.

[33] G. D. Forney, Concatenated Codes. Cambridge, MA: MIT Press, 1966.

[34] H. Hoeve, J. Timmermans, and L. B. Vries, "Error correction in the compact disc system," *Philips Tech. Rev.*, vol. 40, pp. 166-172, Jun. 1982.

[35] G. M. Drury, "DVB channel coding standards for broadcasting compressed video services," *IEE Electron. Commun. Eng. J.*, vol. 9, pp. 11-20, Feb. 1997.

[36] G. D. Forney, "Burst-correcting codes for the classic bursty channel," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 772-781, Oct. 1971.

[37] J. L. Massey, *Threshold Decoding*. Cambridge, MA: MIT Press, 1963.

[38] J. L. Massey and K. K. Sain, "Inverses of sequential circuits," *IEEE Trans. Computers*, vol. C-17, pp. 330-337, Apr. 1968.

[39] J. B. Cain and G. C Clark, Jr., "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding," *IEEE Trans. Info. Theory,* vol. IT-25, pp. 97-100, Jan 1979.

[40] W. W. Wu *et al.*, "Coding for satellite communication," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 724-748, May 1987.

[41] R. Steele, *Mobile Radio Communications*. NY: IEEE Press, 1992.

[42] K. Taura *et al.*, "A digital audio broadcasting (DAB) receiver," *IEEE Trans. Consumer Electron.*, vol 42, pp. 322-327, Aug. 1996.

[43] J. M. Wozencraft, and B. Reiffen, *Sequential Decoding*. Cambridge, MA: MIT Press, 1961.

[44] J. A. Heller, "Feedback decoding of convolutional codes," in *Advances in Commun. Syst.*, A. J. Viterbi, Ed. NY: Academic Press, 1975, pp. 261-278.

[45] J. K. Omura, "On the Viterbi decoding algorithm," *IEEE Trans. Info. Theory*, vol. IT-15, pp. 177-179, Jan. 1969.

[46] G. Fettweis and H Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, pp. 785-790, Aug. 1989.

[47] H. Lou, "Implementing the Viterbi Algorithm," *IEEE Signal Processing Mag.* vol. 12, pp. 42-52, Sep. 1995.

[48] J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 835-848, Oct. 1971.

[49] G. D. Forney, Jr., "Convolutional codes II: Maximum likelihood decoding," *Info. and Control*, vol. 25, pp. 222-266, Jul. 1974.

[50] U. Mengali, R. Pellizzoni, and A. Spalvieri, "Soft-decision-based node synchronisation for Viterbi decoders," *IEEE Trans. Commun.*, vol. 43, pp. 2532-2539, Sep. 1995.

[51] I. Kang and A. N. Wilson Jr., "Low-power Viterbi decoder for CDMA mobile terminals," *IEEE J. Solid-State Circuits*, vol. 33, pp. 473-482, Mar. 1998.

[52] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi decoder VLSI implementation and its performance," *IEEE Trans. Commun.*, vol. 41, pp. 1170-1178, Aug. 1993.

[53] C. B. Shung *et al.*, "VLSI architectures for metric normalisation in the Viterbi algorithm," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Atlanta, GA, Apr. 1990, pp-1723-1728.

[54] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, pp. 1220-1222, Nov. 1989.

[55] T. K. Truong *et al.*, "A VLSI design for a trace-back Viterbi decoder," *IEEE Trans. Commun.*, vol. 40, pp. 616-624, Mar. 1992.

[56] G. D. Forney Jr., "Coset codes: Geometry and classification," *IEEE Trans. Info. Theory*, vol. 34, pp. 1123-1151, Sep. 1988.

[57] B. E. Rimoldi, "A decomposition approach to CPM," *IEEE Trans. Info. Theory*, vol. 34, pp. 260-270, Mar. 1988.

[58] J. K. Wolf and G. Ungerboeck, "Trellis coding for partial-response channels," *IEEE Trans. Commn.*, vol. COM-34, pp. 765-773, Aug. 1986.

[59] H. Kobayashi, "Application of probabilistic decoding to digital magnetic recording systems," *IBM J. Res. Develop.*, vol. 14, pp. 64-74, Jan 1971.

[60] R. Wood, "Magnetic megabits," *IEEE Spectrum*, vol. 27, pp. 32-38, May. 1990.

[61] H. Kobayashi, "Correlative level coding and maximum-likelihood decoding," *IEEE Trans. Info. Theory*, vol. IT-17, pp. 586-594, Sep. 1971.

[62] P. H. Siegel and J. K. Wolf, "Modulation and coding for information storage," *IEEE Commun. Mag.*, vol. 29, pp. 68-86, Dec. 1991.

[63] P. K. D. Pai, A. D. Brewster, and A. A. Abidi, "A 160-MHz analog front-end IC for EPR-IV PRML magnetic storage read channels," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1803-1816, Nov. 1996.

[64] H. K. Thapar and A. M. Patel, "A class of partial-response systems for increasing storage density in magnetic recording," *IEEE Trans. Magn.*, vol. MAG-23, pp. 3666-3668, Sep. 1987.

[65] M. Umemoto, "Interleaved maximum likelihood decoding in high-order partial responses for high-speed magnetic recording systems," in *Proc. IEEE Global Telecom. Conf. (GLOBCOM)*, London, U.K., Nov. 1996, pp. 379-383.

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

[67] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Info. Theory*, vol. IT-9, pp. 64-74, Apr. 1963.

[68] 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.

[69] J. P. Robinson and A. J. Bernstein, "A class of binary recurrent codes with limited error propagation," *IEEE Trans. Info. Theory*, vol. IT-13, pp. 106-113, Jan. 1967.

[70] J. P. Robinson, "Error propagation and definite decoding of convolutional codes," *IEEE Trans. Info. Theory*, vol. IT-14, pp. 121-128, Jan. 1968.

[71] P. Lavoie, D. Haccoun, and Y. Savaria, "New VLSI architectures for fast soft-decision threshold decoders," *IEEE Trans. Commun.*, vol. 39, pp. 200-207, Feb. 1991.

[72] D. Grant, J. Taylor, and P. Houselander, "Design, implementation and evaluation of a high-speed integrated Hamming neural classifier," *IEEE J. Solid-State Circuits*, vol. 29, pp. 1154-1157, Sep. 1994.

[73] C. Verdier, A. Demosthenous, J. Taylor, and M. Wilby, "An integrated analogue convolutional decoder based on the Hamming neural classifier," in *Proc. Neural Networks and their Applications (NEURAP)*, Marseilles, France, Mar. 1996, pp. 150-155.

[74] A. Demosthenous, C. Verdier, and J. Taylor, "A new architecture for low-power analogue convolutional decoders," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Hong Kong, Jun. 1997, pp. 37-40.

[75] P. J. Black and T. H.-Y. Meng, "A 1-Gb/s, four-state, sliding block Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 32, pp. 796-805, Jun. 1997.

[76] S. Kato, M. Morikura, and S. Kubota, "Implementation of coded modems," *IEEE Commun. Mag.*, vol. 29, pp. 88-97, Dec. 1991.

[77] H. Hendrix, "Viterbi decoding techniques in the TMS320C54x family," *Texas Instruments Inc.*, Dallas, Texas, Tech. Rep. SPRA071, Jun. 1996.

[78] J. B. Cain and R. A. Kriete, "A VLSI $R = 1/2$, $K = 7$ Viterbi decoder," in *Proc. Nat. Aerospace Electron. Conf. (NAECON)*, Dayton, OH, May 1984, pp. 20-27.

[79] H. A. Bustamante *et al.*, "Stanford Telecom VLSI design of a convolutional decoder," *IEEE Military Commun. Conf. (MILCON)*, Boston, MA, Oct. 1989, pp. 171-178.

[80] K. Seki *et al.*, "Very low power Viterbi decoder LSIC employing the SSS (scarce state transition) scheme for multimedia mobile communications," *Electron. Lett.*, vol. 30, pp. 637-639, Apr. 1994.

[81] K. Kawazoe *et al.*, "Ultra-high-speed and universal-coding-rate Viterbi decoder VLSIC–SNUFEC," *IEICE Trans. Electron.*, vol. E77-C, no. 12, Dec. 1994.

[82] J. Sparsø *et al.*, "An area-efficient topology for VLSI implementation of Viterbi decoders and other shuffle-exchange type structure," *IEEE J. Solid-State Circuits*, vol. 26, pp. 90-96, Feb. 1991.

[83] H. Dawid, G. Fettweis, and H. Meyr, "A CMOS IC for Gb/s Viterbi decoding: System design and VLSI implementation," *IEEE Trans. VLSI Systems*, vol. 4, pp. 17-31, Mar. 1996.

[84] R. R. Spencer and P. J. Hurst, "Analog implementations of sampling detectors," *IEEE Trans. Magnet.*, vol. 27, pp. 4516-4521, Nov. 1991.

[85] M. H. Shakiba, D. A. Johns, and K. W. Martin, "General approach to implementing analogue Viterbi decoders," *Electron. Lett.*, vol. 30, pp. 1823-1824, Oct. 1997.

[86] A. Demosthenous and J. Taylor, "Current-mode approaches to implementing hybrid analogue/digital Viterbi decoders," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Rhodes, Greece, Oct. 1996, pp. 33-36.

[87] S. Tsukamoto *et al.*, A CMOS 6-b, 200 Msamples/s, 3 V-supply A/D converter for a PRML read channel LSI," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1831-1836, Nov. 1996.

[88] R. C. Davis, "Diode-configured Viterbi algorithm error correcting decoder for convolutional codes," *U.S. Pat.*, no. 4,545,054, Oct. 1, 1985.

[89] T. Suzuki and Y. Yaitoh, "A 100Mbs optical transmission experiment employing a Viterbi decoder composed of analogue circuits," in *Proc. URSI Int. Symp. Signals, Syst. and Electron.*, Erlangen, Germany, 1989, pp. 156-158.

[90] J. B. Hughes and K. W. Moulding, "S^2I switched-current technique for high performance," *Electron. Lett.*, vol.. 29, pp. 1400-1401, Aug. 1993.

[91] EUROPRACTICE IMEC, AMS 0.8μm CMOS Process Parameters, Doc. 9933006 Rev. A.

[92] A. Demosthenous and J. Taylor, "BiCMOS add-compare-select units for Viterbi decoders," due for publication in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Monterey, CA. May. 1998.

[93] D. A. Johns and K. W. Martin, *Analog Integrated Circuit Design*. NY: Wiley, 1997.

[94] A. Demosthenous *et al.*, "Enhanced modular CMOS winner-take-all network," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Rhodes, Greece, Oct. 1996, pp. 402-405.

[95] T. N. Blalock and R.C. Jaeger, "A high-speed clamped bit-line current-mode sense amplifier," *IEEE J. Solid-State Circuits*, vol. 2, pp. 542-548, Apr. 1991.

[96] J. R.-Angulo, "Building blocks for fuzzy processors," *IEEE Circuits and Devices Mag.*, vol. 10, pp. 48-50, Jul. 1994.

[97] I. E. Opris, "Analog rank extractors," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 1114-1121, Dec. 1997.

[98] A. G. Andreou *et al.*, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Trans. Neural Networks*, vol. 2, pp. 205-213, Mar. 1991.

[99] J. B. Hughes and K. W. Moulding, "Switched-current signal processing for video frequencies and beyond," *IEEE J. Solid-State Circuits*, vol. 28, pp. 314-322, Mar. 1993.

[100] Y.-S. Lee and K. W. Martin, "A switched-capacitor realisation of multiple FIR filters on a single chip, " *IEEE J. Solid-State Circuits*, vol. 23, pp. 536-542, Apr. 1988.

[101] S. J. Daubert and D. Vallancount, "A transistor-only current-mode ΣΔ modulator," *IEEE J. Solid-State Circuits*, vol. 27, pp. 821-830, May 1992.

[102] A. Yúfera, A. Rueda, and J. L. Huertas, "Programmable switched-current wave analog filters," *IEEE J. Solid-State Circuits*, vol. 29, pp. 927-935, Aug. 1994.

[103] D. Macq and P. G. A. Jespers, "A 10-bit pipelined switched-current A/D converter," *IEEE J. Solid-State Circuits*, vol. 29, pp. 967-971, Aug. 1994.

[104] T. S. Fiez, G. Liang, D. J. Allstot, "Switched-current design issues," *IEEE J. Solid-State Circuits*, vol. 26, pp.192-202, Mar. 1991.

[105] G. Wegmann and E. A. Vittoz, "Analysis and improvements of accurate dynamic current mirrors," *IEEE J. Solid-State Circuits*, vol. 25, pp. 699-706, Jun. 1990.

[106] S. J. Daubert, D. Vallancourt, and Y. P. Tsividis, "Current copier cells," *Electron. Lett.,* vol. 24, pp. 1560-1562, Dec. 1988.

[107] P. M. Sinn and G. W. Roberts, "A comparison of first and second generation switched-current cells," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, London, U.K., May 1994, pp. 301-304.

[108] C. Toumazou, J. B. Hughes and D. M. Pattullo, "Regulated cascode switched-current memory cell," *Electron. Lett.*, vol. 26, pp. 303-305, Mar. 1990.

[109] W. Guggenbühl, J. Di, and J Goette, "Switched-current memory cells for high-precision applications," *IEEE J. Solid-State Circuits*, vol. 29, pp. 1108-1116, Sep. 1994.

[110] D. G. Nairn, "A high-linearity sampling technique for switched-current circuits," *IEEE Trans. Circuits Syst. II*, vol. 49-52. Jan. 1996.

[111] C. Eichenberger and W. Guggenbühl, "On charge injection in analog MOS switches and dummy switch compensation techniques," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 256-264, Feb. 1990.

[112] M. Song, Y. Lee, and W. Kim, "A clock feedthrough reduction circuit for switched-current systems," *IEEE J. Solid-State Circuits*, vol 28, pp. 133-137, Feb. 1993.

[113] D. W. J. Groeneveld *et al.*, "A self-calibration technique for monolithic high-resolution D/A converters," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1517-1522, Dec. 1989.

[114] R. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust. Speech Signal Processing Mag.*, vol. 4, pp. 4-22, Apr. 1987.

[115] M. E. Robinson, H. Yoneda, and E S.-Sinencio, "A modular CMOS design of a Hamming network," *IEEE Trans. Neural Networks*, vol. 3, pp. 444-456, May 1992.

[116] W.-C. Fang *et al.*, "A VLSI neural processor for image data compression using a self-organising network," *IEEE Trans. Neural Networks*, vol. 3, pp. 506-518, May 1992.

[117] G. T. Tuttle, S. Fallahi, and A. A. Abidi, "An 8-b CMOS vector A/D converter," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, Feb. 1993, pp. 38-39.

[118] G. Gauwenberghs and V. Pedroni, "A low-power CMOS analog vector quantiser," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1278-1283, Aug. 1997.

[119] J. E. Fowler, Jr. *et al.*, "Real-time video compression using differential vector quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 14-24, Feb. 1995.

[120] A. K. Jain, "Image data compression: A review," *Proc. IEEE*, vol. 69, pp. 349-389, Mar. 1981.

[121] C.-L. Wang and K.-M Chen, "A new VLSI architecture for full-search vector quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 389-398, Aug. 1996.

[122] A. K. Krishnamurthy *et al.*, "Neural networks for vector quantization of speech and images," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1449-1457, Oct. 1990.

[123] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.* Vol. 36, pp. 957-971, Aug. 1988.

[124] Y. He and E. S.-Sinencio, "Min-net winner-take-all CMOS implementation," *Electron. Lett.*, pp. 1237-1239, Jul. 1993.

[125] U. Çilingiroglu, "A charge-based neural Hamming classifier," *IEEE J. Solid-State Circuits*, vol. 28, pp. 59-67, Jan. 1993.

[126] S. Smedley, J. Taylor, and M. Wilby, "A scalable high-speed current-mode winner-take-all network for VLSI neural applications," *IEEE Trans. Circuits Syst.* I, vol. 42, pp. 289-291, May 1995.

[127] T. S.-Gotarredona and B. L.-Barranco, "A high-precision current-mode WTA-MAX circuit with multichip capability," *IEEE J. Solid-State Circuits*, vol. 33, pp. 280-280, Feb. 1998.

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

[129] V. A. Pedroni, "Inhibitory Mechanism analysis of complexity $O(N)$ MOS winner-take-all networks," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 172-175, Mar. 1995.

[130] M. Sasaki et al., "Fuzzy multiple-input maximum and minimum circuits in current mode and their analyses using bounded-difference equations," *IEEE Trans. Comput.*, vol.39, pp.768-774, Jun. 1990.

[131] C. Das, "MIETEC 2.4μm CMOS MPC–Electrical parameters," *EUROCHIP Service Org. (RAL)*, Didcot, U.K., Tech. Rep. MIE/F/02, Jan. 1993.

[132] A. Demosthenous and J. Taylor, "A winner-take-all network for large-scale analogue vector quantisers," due for publication in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Monterey, CA, May. 1998.

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