University College London

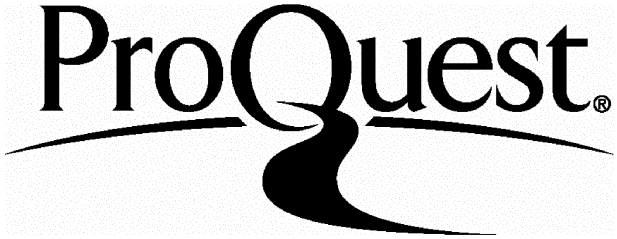
Novel Architectures for Forward Error Control at Very High Bit Rates

Thesis Submitted in Candidature for the Degree of
Doctor of Philosophy

September 1997

Richard Spencer Blake

Department of Electrical and Electronic Engineering
University College London
London
United Kingdom



Statement of Originality

The work presented in this thesis was carried out by the candidate . It has not been presented previously for any degree, nor is it at present under consideration by any other degree awarding body.

Candidate:

Richard Spencer Blake.

Director of Studies:

Professor John J. O'Reilly.

Statement of Availability

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Candidate:

Richard Spencer Blake.

Acknowledgements

I wish to extend my sincerest thanks to Professor John O'Reilly for his guidance and encouragement during the period of my studies. I would also like to thank him for his help in the preparation of my thesis.

I am also grateful to Dr Andy Popplewell for his informative comments and wealth of knowledge which proved invaluable during the early stages of this work.

Financial support for this research has been provided by the Physical and Engineering Research Council, BT Laboratories Martlesham and Professor John O'Reilly, for which I am very grateful.

Finally I would like to give a special mention to my mother Elizabeth, who has supported and encouraged me throughout my time in academia and Jane for believing in me and above all putting up with me - thank you both.

Summary

This thesis is concerned with the design and assessment of novel architectures for the implementation of forward error correction (FEC) coding systems for very high bit rate operation. This is motivated by both the development of multi giga-bit (especially optical) transmission systems and the increasingly demanding error performance targets for digital telecommunications. It is shown that present serial FEC architectures are often inadequate for demanding high bit rate applications and can only achieve high data rates by separately encoding and then multiplexing several tributary data streams. Alternatively parallel encoders, whilst offering the prospect of increased operational speed, are often far too complex for all but the most trivial of codes.

To overcome these limitations series-parallel FEC techniques - derived from earlier work on m-sequence generation - are introduced and examined. By describing the functional specification of the encoding and error detection circuits in the form of a transition matrix it is possible, by matrix manipulation, to define alternative circuits which allow a trade off between circuit speed and complexity.

Having demonstrated how series-parallel techniques may be applied to high speed encoding and error detection attention is then focused on error correction. By taking advantage of transmission channel statistics and using high speed error detection, a buffered decoding arrangement is explored which is shown to operate at an average, rather than the worst case, speed. This decoder, used in conjunction with series-parallel encoding and error detection circuits, can provide the basis for the realisation of a complete high speed FEC system.

The thesis then concludes with an illustrative case study concerning the benefits of employing FEC to a new generation of long haul optically amplified submarine systems. Currently proposed error control strategies are reviewed; a comparison is effected with low complexity binary BCH codes which may be realised at the system line rate using the architectures and arrangements developed in this thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Background

Digital information transmission is now universally employed for long distance telecommunications [1]. Over the past two decades technical advances in this field have resulted in many new techniques and standards for digital transmission being developed [2-4]. As consumer demand for greater capacity and improved reliability of digital networks increases, this trend looks set to continue well into the next century.

An area which has benefited significantly from this continual research is fibre optic transmission systems. As device technology matured transmission bandwidths increased considerably from early systems operating at up to 140 Mbit/s to current systems operating at 2.5 Gbit/s and beyond [5]. As the boundaries of system performance are extended, with current experimental systems demonstrating capacities of 100 Gbit/s [6], the various transmission impairments encountered become increasingly significant.

At the same time, there have been significant advances in digital signal processing technology and in particular in forward error correction (FEC). Also there have been preliminary efforts to apply FEC as a means to ameliorating impairments in very long haul, high capacity optical fibre transmission systems, such as those used for trans-oceanic communications [7-12]. This makes it appropriate to study more generally the extent to which FEC may be adapted to meet the requirements of high data rate telecommunication transmission.

Accordingly, this research is concerned with the application and implementation of various coding schemes, placing emphasis on generic architectures appropriate to very high speed operation and on their attendant performance. As a particular application area, consideration is given to the requirements for a new generation of lightwave submarine transmission systems which exploit optical amplifiers.

1.2 Thesis organisation

Following this brief introduction chapter 2 provides a review of aspects of error control coding theory. Here the fundamental structure and algebraic nature of both binary and non-binary cyclic codes is examined. In particular attention is drawn to a class of powerful yet relatively simple random error correcting codes known as BCH codes. The chapter concludes with a brief overview of decoding methods using syndromes and acknowledges various error correction schemes based on time domain methods.

Building upon this, chapter 3 then illustrates common encoding arrangements based on both the generator and parity polynomial of a BCH code. In each case generalised examples of fully serial and fully parallel encoders are given which again relate

to both binary and non-binary codes. The limitations of each arrangement are then discussed before concluding the chapter with serial and parallel error detection circuits based on minimal polynomials.

Having discussed the major limitations of conventional encoding/error detection circuitry, chapter 4 introduces series-parallel architectures which provide a trade off between the high speed of a parallel arrangement and the low complexity of a serial arrangement. Viewing the network of feedback shift registers as an autonomous circuit, a transition matrix may be defined which describes the functional specification of the circuit. Subsequent processing of this matrix allows intermediate circuit solutions to be generated which incorporate varying degrees of parallelism into the design of the encoder or error detector.

Although high speed encoding and error detection has been demonstrated in chapter 4 it is found that decoding, or more specifically error correction, proves to be more problematic. Chapter 5 examines how high speed decoding may be realised using buffered decoding techniques. By taking advantage of channel error statistics and high speed error detection circuits a decoding system may be realised which operates at speeds comparable to those of the encoding architectures presented in chapter 4. This is done with a view to providing a complete high speed error control system capable of operating in the multi giga-bit region.

In chapter 6 forward error control coding is considered in the context of long haul optically amplified submarine systems where transmission rates in excess of 2.5 Gb/s are found. Comparing currently proposed error control strategies with well established low complexity binary BCH codes offers the prospect of realising forward error control at the line rate by using the structures presented in the previous chapters. Furthermore, by encoding and decoding at the line rate the code may be potentially matched to anticipated channel error patterns associated with known

error inducing mechanisms and signal conditions.

Finally, chapter 7 concludes the thesis by summarising the main findings of this research and provides suggestions for further work.

1.3 Summary of Main Contributions

The research presented in this thesis investigates the possibility of implementing novel architectures to realise forward error control coding arrangements for application in long haul optically amplified submarine systems. The major contributions resulting from this work may be summarised as follows:

- Standard architectures used for encoding and error detection were evaluated and shown to be either too complex or too slow for modern high speed digital communication systems.
- The use of series-parallel architectures applied to error control circuitry indicated a trade off between circuit speed and complexity. Speeds in the giga-bit per second region may be achieved by these methods even when conventional logic families such as ECL and CMOS are used.
- Buffered decoding techniques have demonstrated the ability to provide an overall enhancement in decoding speed at the expense of decoding delay.
- Implementation of a buffered decoder using a standard RS decoding IC as the main error correction element has been investigated and shown to offer an increase in speed over stand alone devices. Further speed increases may be achieved if the decoder is optimised for certain error patterns.

- The use of low complexity BCH codes has been shown to offer comparable performance to that of multi-level Reed-Solomon codes in the presence of random errors. This offers the prospect of encoding and decoding at the line rate thereby allowing a certain degree of control over the transmitted signal.

The contributions made during the course of this research have led to the following publications to date:

1. J.J. O'Reilly, A. Popplewell and R. Blake. 'Forward error control for international telecommunications transmission'. IEE Colloquium on International Transmission Systems, pp. 8/1-8/4, February 1994.

2. Y. Bian, R. Blake, A. Popplewell, J. O'Reilly and S. Fragiacomo. 'FEC for Future Trans-Oceanic Optical Systems', Fifth IEE Conference on Telecommunications, Brighton, pp 78-82, March 1995.

3. J J O'Reilly and R S Blake. 'Novel coding techniques for long-haul high capacity optical transmission systems'. Invited Presentation at 1st National Telecommunications Conference, Aveiro, Portugal, April 1997.

1.4 Summary

This chapter has presented the motivation and background for investigating novel architectures for the implementation of forward error control codes in the context of high speed digital communications systems. Having outlined the structure of the thesis and provided a summary of the main contributions we now move on, in chapter 2, to a general introduction to error control codes.

Chapter 2

Error Control Coding

2.1 Introduction

The theory of error control coding has long been established, in fact its origins may be traced back to the pioneering work of Shannon in 1948 [13] [14]. The 1950s and 60s saw many advances in this field with much time and effort being devoted to the formulation of new codes and coding techniques.

The rapid growth of digital technology over recent decades has seen the emphasis shift from a theoretical nature to a more applications-focussed approach. As a result Error Control Codes (ECCs) have now found many applications in field of digital communications and storage. Such applications range from satellite communications and computer disk drives to compact discs and more recently optical transmission systems.

In order to provide a foundation for following work this chapter will introduce and review some simple but none the less very important concepts of error control coding.

Starting with the basic ideas of error control coding we move on to review some of the properties associated with ECCs. The nature of cyclic codes is then considered along with the mathematical structure of a class of random ECCs known as BCH codes. The chapter concludes by discussing the various aspects and strategies associated with decoding. In particular, methods for error detection/correction are highlighted.

2.2 Error Control Coding

Simply stated, error control coding may be defined as the mapping of one data sequence onto another, thereby improving the overall reliability of the channel over which the data is transmitted. The process of error control coding involves the introduction of redundant information into the transmitted signal, thereby enabling error detection and/or error correction to be performed at the decoder. The manner in which this mapping occurs often defines the properties and nature of the code.

In general there are two main types of error control coding system, (i) Automatic repeat Request (ARQ) and (ii) Forward Error Correction (FEC). ARQ systems rely on the receiver or decoder to detect the presence of errors in the received codewords and then, via a feedback path, request re-transmission of any erroneous data. FEC systems however, are not only able to detect the presence of errors, but are structured in such a manner that error correction may be performed by the decoder thereby circumventing the need for a feedback path. It is often convenient, under certain circumstances, to combine these two schemes to produce a third option known as hybrid-ARQ. Although ARQ and hybrid-ARQ systems have their respective benefits, the following work will only concern itself with FEC. The prevalence of ARQ systems and their relation to high speed transmission systems will be discussed in a later chapter.

2.3 Classification of error control codes

The phrase "error control coding" is often used to describe a broad spectrum of coding topics. In order to differentiate between the various types of code it is appropriate to define the following coding hierarchy.

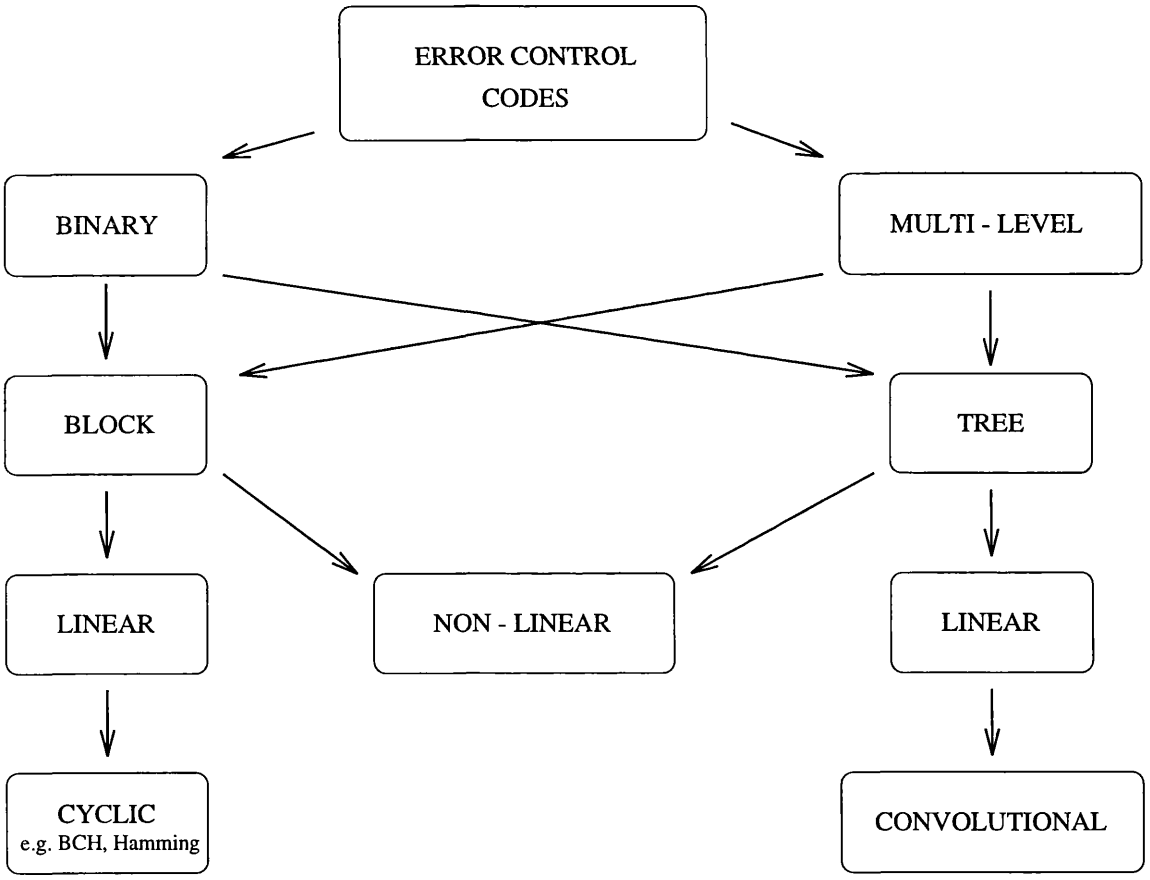


Figure 2.1: Hierarchy of error control codes.

It can clearly be seen that there exist two fundamental types of code, binary and non-binary or multi-level codes. A code is termed binary if the symbols used to generate its codewords are defined over a binary alphabet i.e. 0s and 1s, however if an alternative alphabet is used then the code is described as multi-level.

Both binary and multi-level codes can be further subdivided into block and tree codes. Tree codes may be distinguished from block codes as their codewords are generated from not only the current information word presented to the encoder but also the previous m transmitted words, where m is the memory of the encoder. In addition both block and tree codes may be partitioned into linear and non-linear codes. An important class of linear tree code which is commonly used in the field of telecommunications is the convolutional code. This type of code is often employed in mobile telephony and satellite systems [15].

For the purposes of this work attention will be directed towards a subset of binary and non-binary block codes called linear cyclic codes. In particular a class of cyclic codes known as (BCH) codes will be examined in both binary and non-binary form.

2.4 Block Codes

Block coding is a mathematical method of mapping a block of k information or source symbols onto a block of n codeword symbols using a pre-defined algorithm. The encoding process divides the message sequence into blocks of k symbols represented by the k-tuple vector $\mathbf{u}=[u_0, u_1, ..., u_{k-1}]$. Each message block is then transformed into a corresponding codeword of n discrete symbols denoted by the n-tuple vector $\mathbf{v}=[v_0, v_1, ..., v_{n-1}]$. The result is an (n, k) block code with the inserted $n-k$ elements commonly known as parity check symbols or digits. For an (n, k) code with symbols from the Galois Field GF(2^m) there exist $(2^m)^k$ possible messages (for binary codes where m=1 there are just 2^k possible messages). Figure 2.2 illustrates the encoding procedure.

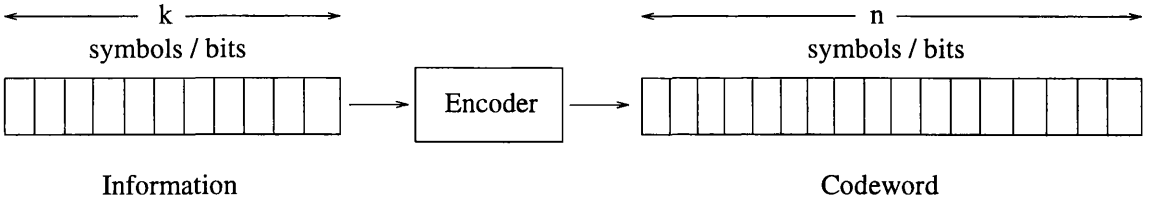


Figure 2.2: Encoding procedure.

As previously stated there are two main types of code to be considered, binary and non-binary. For simplicity the following theory and definitions will be restricted to binary codes, although it may easily be generalised and adapted to non-binary cases.

Definition 2.1

A block code of length n and 2^k codewords is called a linear (n, k) code, $\mathcal{C}$, if and only if its 2^k codewords form a k-dimensional subspace of the vector space of all the n-tuples over the Galois Field GF(2).

The implied algebraic structure of these linear codes together with definition 2.1 suggests that codewords may be generated from a set of k basis codewords. This property is of great practical significance as it enables long and complex codes to be generated with only modest resources.

Definition 2.2

The rate of a block code is defined as the ratio of the number of information symbols to that of the number of codeword symbols i.e.

$$\text{Rate} = \frac{k}{n} \tag{2.1}$$

This is an important metric as it defines the amount of redundancy introduced into the transmitted data by the encoding operation. High rate codes are favourable for two reasons: In the first instance if the transmission rate is limited then the amount

of redundant data transmitted is kept to a minimum; secondly, for a constant data rate i.e. increased transmission rate, the noise penalty due to increased bandwidth is minimised.

Definition 2.3

The Hamming weight, $w(\mathbf{v})$, or weight of a codeword $\mathbf{v}$ is defined as the number of non-zero elements of that codeword e.g. the codeword $\mathbf{v}$=1001100 has $w(\mathbf{v})$=3.

Definition 2.4

The distance or Hamming distance d($\mathbf{v}$,$\mathbf{w}$) between two codewords, $\mathbf{v}$ and $\mathbf{w}$, is the number of positions in which they differ. This result may be directly obtained by the modulo 2 addition of the two codewords.

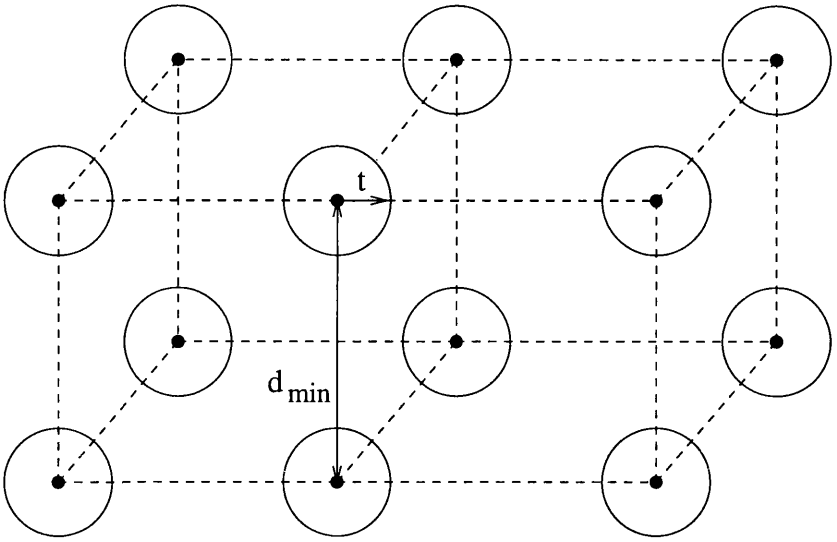
Definition 2.5

The minimum distance d_{min}, of a code $\mathcal{C}$, is defined as the smallest modulo 2 sum of any two codewords and it can be shown that this is equal to the minimum weight of the codeword set:

i.e. $$d_{min} = \min\{\ d(\mathbf{v},\mathbf{w}) : \mathbf{v},\mathbf{w} \in \mathcal{C}, \mathbf{v} \neq \mathbf{w}\ \}$$

Figure 2.3 symbolically illustrates the properties of a t error correcting block code. Each codeword is surrounded by a sphere of radius t. Contained within each sphere is not only a valid codeword but numerous n-tuple vectors which differ from the codeword in up to t places. Assuming that the error correcting capabilities of the code are not exceeded any received vector lying within a sphere will be correctly decoded as the codeword associated with that sphere. This is known as minimum distance decoding and it is this property that forms the basis of many error correction schemes. To ensure correct decoding no spheres may overlap, therefore for a t error

correcting code the distance between any two spheres must be greater than $2t$ i.e. $d_{min} \geq 2t + 1$.



• Codeword

Figure 2.3: Symbolic representation of a codeword set.

2.4.1 Cyclic Codes

Cyclic codes represent an important subclass of linear codes. Their cyclic nature allows encoding and error detection to be easily implemented by a network of Linear Feedback Shift Registers (LFSRs). The inherent algebraic structure of these codes results in various practical methods being used to decode them.

If the components of a codeword are $\mathbf{v} = (v_0, v_1, ..., v_{n-1})$ then cyclically shifting each component one place to the right results in the n-tuple,

$$\mathbf{v}^{(1)} = (v_{n-1}, v_0, v_1, ..., v_{n-2})$$

which is called a cyclic shift of $\mathbf{v}$.

In general if the components of $\mathbf{v}$ are shifted i places to the right then the resulting n-tuple is

$$\mathbf{v}^{(i)} = (v_{n-i}, v_{n-i+1}, \ldots, v_{n-1}, v_0, v_1, \ldots, v_{n-i-1}).$$

It can be clearly seen that cyclically shifting a codeword i places to the right is equivalent to cyclically shifting the same codeword $n - i$ places to the left.

Definition 2.6

An (n, k) linear code is called a cyclic code if every cyclic shift of a code vector in $\mathcal{C}$ is also a code vector in $\mathcal{C}$.

2.4.2 Systematic Codes

It is often convenient from a design point of view to produce codes in a systematic format as shown in figure 2.4. The codeword is divided into two parts, a message part and a redundant parity check part. The message part consists of the k unaltered information symbols while the redundant parity check part consists of the $(n - k)$ parity check symbols, which are linear sums of the information symbols.

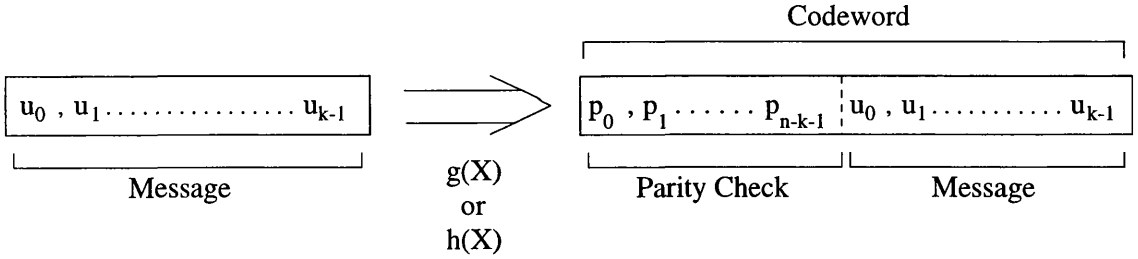


Figure 2.4: Systematic format of a codeword.

2.5 Polynomial Representation of Codes

For encoding and decoding purposes it is often useful to represent a code in polynomial form. The codeword vector $\mathbf{v}=[v_0, v_1, ..., v_{n-1}]$ is expressed as a polynomial of degree $n-1$ or less with the coefficients of X^j obtained from the j^{th} component of $\mathbf{v}$

i.e.

$$\boldsymbol{v}(X) = v_0 + v_1X^1 + v_2X^2 + ... + v_{n-1}X^{n-1} \tag{2.2}$$

Similarly the information vector $\mathbf{u}=[u_0, u_1, ..., u_{k-1}]$ can be represented as a polynomial of degree $k-1$ or less thus

$$\boldsymbol{u}(X) = u_0 + u_1X^1 + u_2X^2 + ... + u_{k-1}X^{k-1} \tag{2.3}$$

2.6 BCH Codes

One of the most important and powerful classes of linear block codes are BCH codes. Binary BCH codes were discovered by Hocquenghem in 1959 [16] and independently by Bose and Chaudhuri in 1960 [17]. In 1961 Gorenstein and Zierler [18] generalised these results to include codes with symbols from GF(p^m) where p is a prime. An important class of non-binary BCH codes are the Reed-Solomon (RS) codes [19].

BCH codes represent some of the most extensively studied random error correcting codes. Many good encoding and decoding arrangements have been devised over the years which take advantage of the highly algebraic structure and cyclic nature of these codes. It is appropriate here to review some of the more important properties of both binary and Reed-Solomon BCH codes.

2.6.1 Binary Codes

The most common BCH codes, known as primitive BCH codes, may be characterised as follows. For any given positive integer $m \geq 3$ and $t \leq 2^{m-1}$ there exists a binary BCH code with the following parameters:

Block Length:	$n = 2^m - 1$
Number of parity-check digits:	$n - k \leq mt$
Minimum distance:	$d_{min} \geq 2t + 1$

A BCH code is uniquely defined in terms of its generator polynomial or parity polynomial. The generator polynomial for a t-error-correcting code of length $n = 2^m - 1$ is specified in terms of its roots from the Galois field GF(2^m). If α is a primitive element in GF(2^m) then $\boldsymbol{g}(X)$ is the lowest degree polynomial which has

$$\alpha^1, \alpha^2, \alpha^3, .., \alpha^{2t} \tag{2.4}$$

as its roots. Let $\boldsymbol{\Phi}_i$ be the minimal polynomial which has α^i as a root. Then $\boldsymbol{g}(X)$ is the lowest common multiple of $\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, ..., \boldsymbol{\Phi}_{2t}$ i.e.

$$\boldsymbol{g}(X) = \text{LCM}\{\boldsymbol{\Phi}_1(X)\boldsymbol{\Phi}_2(X)\boldsymbol{\Phi}_3(X), ..., \boldsymbol{\Phi}_{2t}\} \tag{2.5}$$

It has been shown [20] that every even power of α in the sequence 2.4 has the same minimal polynomial as some preceding odd power of α. The generator polynomial uniquely describing a code can therefore be modified to:

$$\boldsymbol{g}(X) = \text{LCM}\{\boldsymbol{\Phi}_1(X)\boldsymbol{\Phi}_3(X)...\boldsymbol{\Phi}_{2t-1}(X)\} \tag{2.6}$$

With the resulting generator taking the form

$$\boldsymbol{g}(X) = g_0 + g_1X + g_2X^2 + ... + g_{n-k-1}X^{n-k-1} + X^{n-k}. \tag{2.7}$$

Alternatively a code may be uniquely specified by its parity polynomial $\boldsymbol{h}(X)$. For an (n, k) linear code this is related to the generator polynomial by the following

equation

$$\boldsymbol{h}(x) = \frac{X^n + 1}{\boldsymbol{g}(X)} \tag{2.8}$$

where $\boldsymbol{h}(X)$ is a polynomial of degree k with the following form:

$$\boldsymbol{h}(X) = h_0 + h_1 X + h_2 X^2 + ... + h_{k-1} X^{k-1} + X^k. \tag{2.9}$$

In both cases we may note that the coefficients of $\boldsymbol{g}(X)$ and $\boldsymbol{h}(X)$ are defined over GF(2) for a binary code. As an example consider the (15,7) double error correcting code. Constructing the Galois Field GF(2^4) from the primitive polynomial $1 + X + X^4 = 0$ and using α as a primitive element, we find the minimal polynomials which have α and α^3 as roots as

$$\begin{aligned} \boldsymbol{\Phi}_1(X) &= 1 + X + X^4 \\ \boldsymbol{\Phi}_3(X) &= 1 + X + X^2 + X^3 + X^4 \end{aligned}$$

Sine the generator polynomial is defined as

$$\boldsymbol{g}(X) = \text{LCM}\{\boldsymbol{\Phi}_1(X)\boldsymbol{\Phi}_3(X)\}.$$

and $\boldsymbol{\Phi}_1(X)$ and $\boldsymbol{\Phi}_3(X)$ are in this case distinct irreducible polynomials,

$$\begin{aligned} \boldsymbol{g}(X) &= \boldsymbol{\Phi}_1(X)\boldsymbol{\Phi}_3(X) \\ &= (1 + X + X^4)(1 + X + X^2 + X^3 + X^4) \\ &= 1 + X^4 + X^6 + X^7 + X^8 \end{aligned}$$

and the parity polynomial for the (15,7) code may be calculated thus

$$\begin{aligned} \boldsymbol{h}(X) &= \frac{X^{15} + 1}{(1 + X^4 + X^6 + X^7 + X^8)} \\ &= 1 + X^4 + X^6 + X^7. \end{aligned}$$

To illustrate the encoding process we shall consider a simple (7,4) Hamming code based on the generator polynomial $\boldsymbol{g}(X) = 1 + X + X^3$. Suppose the information

sequence to be encoded is 0 1 1 1, represented as a polynomial $\boldsymbol{u}(X) = X + X^2 + X^3$. Encoding in a non-systematic form simply involves multiplying the information polynomial $\boldsymbol{u}(X)$ by the generator polynomial $\boldsymbol{g}(X)$ using binary modulo-2 addition and multiplication.

Message $\boldsymbol{u}(X)$	Encoding	Codeword $\boldsymbol{v}(X)$
0 1 1 1	$\boldsymbol{u}(X).\boldsymbol{g}(X)$	0 1 0 0 0 1 1
$X + X^2 + X^3$	$(X + X^2 + X^3).(1 + X + X^3)$	$X + X^5 + X^6$

Where . denotes multiplication.

Table 2.1 shows an example of a (7,4) code in both systematic and non-systematic form. It can be clearly seen that for both cases the same codeword set is used, the only difference occurring in the mapping between the information vector and the codeword. For any given information vector a systematic codeword may easily be generated by simply selecting the non-systematic codeword which contains the corresponding k information bits in the highest order positions of the codeword.

Information	Codeword	
	Non-Systematic	Systematic
0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0
0 0 0 1	0 0 0 1 1 0 1	1 0 1 0 0 0 1
0 0 1 0	0 0 1 1 0 1 0	1 1 1 0 0 1 0
0 0 1 1	0 0 1 0 1 1 1	0 1 0 0 0 1 1
0 1 0 0	0 1 1 0 1 0 0	0 1 1 0 1 0 0
0 1 0 1	0 1 1 1 0 0 1	1 1 0 0 1 0 1
0 1 1 0	0 1 1 0 1 0 0	1 0 0 0 1 1 0
0 1 1 1	0 1 0 0 0 1 1	0 0 1 0 1 1 1
1 0 0 0	1 1 0 1 0 0 0	1 1 0 1 0 0 0
1 0 0 1	1 1 0 0 1 0 1	0 1 1 1 0 0 1
1 0 1 0	1 1 1 0 0 1 0	0 0 1 1 0 1 0
1 0 1 1	1 1 1 1 1 1 1	1 0 0 1 0 1 1
1 1 0 0	1 0 1 1 1 0 0	1 0 1 1 1 0 0
1 1 0 1	1 0 1 0 0 0 1	0 0 0 1 1 0 1
1 1 1 0	1 0 0 0 1 1 0	0 1 0 1 1 1 0
1 1 1 1	1 0 0 1 0 1 1	1 1 1 1 1 1 1

Table 2.1: Systematic and Non-systematic form of codewords

2.6.2 Non-Binary Codes

The most important sub-class of non-binary BCH codes are the Reed-Solomon or RS codes. Named in honour of their discoverers, RS codes represent a class of highly efficient random error correcting codes with excellent burst error correcting capabilities when transmitted over binary channels.

The encoding and decoding of RS codes differs from binary codes in that all mathematical operations are performed on symbols rather than individual bits. Specifically, an (n, k) RS code maps a block of k information symbols into a block of n codeword symbols, with each symbol being represented by m bits, where $m \geq 1$.

For a t-error correcting code with symbols from GF(2^m) there exists an RS code with the following parameters:

Block Length:	$n = 2^m - 1$
Number of parity-check digits:	$n - k = 2t$
Minimum distance:	$d_{min} = 2t + 1$

We may note that the length of the code is one less than the radix of a code symbol and the minimum distance is one greater than the number of parity check symbols. The highly efficient use of redundancy compared with binary codes and their flexibility, in terms of block length and symbol size, makes RS codes very attractive for commercial use.

Like the binary code the RS code is specified in terms of its roots defined over GF(2^m). However the generator polynomial of a t error correcting RS code is defined

as:

$$\begin{aligned} \boldsymbol{g}(X) &= \sum_{i=1}^{2t}(X+\alpha^i) \\ &= g_0 + g_1X + g_2X^2 + ... + g_{2t-1}X^{2t-1} + X^{2t}. \end{aligned} \tag{2.10}$$

Where α is a primitive element in GF(2^m). The parity polynomial of an RS code is derived in the same way to that of the binary BCH codes.

Definition 2.7

A cyclic code $\mathcal{C}$ is constructed by multiplying a generator polynomial $\boldsymbol{g}(X)$ by all polynomials of degree $k-1$ or less.

2.7 Decoding of Linear Block Codes

The decoding of linear block codes may be regarded as three discrete operations; error detection, error correction and the recovery of the message or information sequence. For codewords in systematic form the recovery of the information becomes a trivial process as the message is present and unaltered in the transmitted codeword. It is therefore only necessary to discard the $(n-k)$ parity check digits once any error correction has taken place. Of the two remaining operations error detection is by far the simplest. This involves calculating a syndrome vector in order to determine if the received vector, $\boldsymbol{r}(X)$, is a valid codeword. For an (n,k) cyclic code where $n = 2^m - 1$ the syndrome is represented by a vector containing $2t$ m-tuples thus $\mathbf{S}=\{S_1, S_2, ..., S_{2t}\}$. The components of $\mathbf{S}$ are defined as

$$S_i = \boldsymbol{r}(\alpha^i) \tag{2.11}$$

for $1 \leq i \leq 2t$ where $\boldsymbol{r}(\alpha^i)$ is the received vector evaluated at α^i. The received vector may be represented as a linear combination of a valid codeword and error vector

$\boldsymbol{e}(X)$ thus

$$\boldsymbol{r}(X) = \boldsymbol{c}(X) + \boldsymbol{e}(X). \tag{2.12}$$

By definition 2.7 each codeword is a multiple of $\boldsymbol{g}(X)$ and therefore must also be a multiple of $\boldsymbol{\Phi}_i(X)$ for $1 \leq i \leq 2t$ (see equation (2.5)). Equation (2.12) can now be written as

$$\boldsymbol{r}(X) = \boldsymbol{a}(X)\boldsymbol{\Phi}_i(X) + \boldsymbol{e}(x) \tag{2.13}$$

for $1 \leq i \leq 2t$. Substituting values of α^i gives

$$\boldsymbol{r}(\alpha^i) = \boldsymbol{e}(\alpha^i) \tag{2.14}$$

since $\boldsymbol{\Phi}_i(\alpha^i)$=0. As the syndrome component S_i is defined as $\boldsymbol{r}(\alpha^i)$ we obtain the following

$$S_i = \boldsymbol{e}(\alpha^i). \tag{2.15}$$

That is to say, the syndrome components are dependent only on the error pattern and not the transmitted codeword. Clearly $\mathbf{S}=\underline{0}$ if and only if $\boldsymbol{r}(X)$ is a codeword, and $\mathbf{S}\neq \underline{0}$ if and only if $\boldsymbol{r}(X)$ is not a codeword. Therefore, when $\mathbf{S}\neq \underline{0}$ we know that $\boldsymbol{r}(X)$ is not a codeword and the presence of errors has been detected. However, if the error pattern is such that it transforms the transmitted codeword into another valid codeword i.e. the error pattern is itself a valid codeword then $\mathbf{S}=\underline{0}$. Error patterns of this kind are called undetectable error patterns and cause decoding errors.

In contrast to error detection, error correction proves to be somewhat more involved. Many good algorithms exists for error correction [20-23], nearly all of which employ information gained from the syndromes. One possible strategy involves storing the syndromes associated with every correctable error pattern. By matching the syndrome of a received codeword to one that is already stored it is possible to find the error pattern. Unfortunately for large codes or codes which are capable of correcting

a large number of errors this option is not feasible. Alternatively, the error pattern may be calculated by another method. One such method is the Berlekamp-Massey algorithm [24] which involves using finite field arithmetic in order to solve a set of simultaneous equations and calculate the roots of an error locator polynomial. Although relatively simple in theory the practical requirements of such a method often demand considerable complexity. This has lead to many coding theorists developing alternative sub-routines in an attempt to reduce complexity. Many other solutions exist which take advantage of the cyclic nature of BCH codes [25] [26]. For example an error trapping decoder [27] does not directly calculate the error locations but assumes that errors occur in specific patterns. As a result a decoder of reduced complexity, and often increased speed, can be implemented. Therefore it is often the application and implementation factors which dictate what type of decoding arrangement is required.

2.8 Summary

In summary the process of error control coding introduces redundancy into the message sequence to enable error detection and/or error correction. The mapping between the information sequences and codewords is unique and is performed over a finite field. The algebraic structure of the codes often makes the encoding and error detection process a simple matter. However, error correction is usually more complex but can, with careful planning, be achieved with relative ease.

We now continue, in the next chapter, by examining some standard architectures for the encoding and error detection of BCH codes. We then proceed by introducing some new concepts which will effect a considerable increase in speed.

Chapter 3

Standard Encoding and Error Detection Methods

3.1 Introduction

Following the general review of ECC in the last chapter a brief appraisal of well established techniques devised for the realisation of encoding and error detection systems is now given. Current serial architectures based on both the generator and parity polynomial of a code will be reviewed and the attributes of each method discussed. In addition, parallel encoding arrangements will be introduced and shown to provide a substantial gain in operational speed at the cost of greatly increased circuit complexity.

Having examined various encoding arrangements, error detection circuits based on minimal polynomials are then presented. After noting the similarities to encoding architectures it is shown that both serial and parallel configurations are subject to similar speed/complexity constraints.

3.2 Standard Encoding Architectures

The implementation of cyclic block codes has traditionally involved the use of Linear Feedback Shift Register (LFSR) networks to achieve polynomial division or multiplication. This section introduces and briefly reviews standard architectures used for the generation of BCH codes.

Initially, common serial encoding arrangements derived from the generator and parity polynomials of an arbitrary code will be considered. For each arrangement an illustrative example, based on a binary Hamming code, will be given. After highlighting the key differences which exist between each method, the respective merits and disadvantages of the two arrangements are discussed. An alternative solution to the problem of codeword generation is then introduced. The direct extraction of the parity check bits via the parity check equations offers the prospect of high speed encoding at the expense of greatly increased circuit complexity.

3.2.1 Serial Encoding

A method often employed for the generation of BCH codewords is one based on algebraic polynomial division or multiplication. By exploiting the strong algebraic structure of cyclic codes it is possible to implement an encoder based on a appropriately configured network of feedback shift registers. Circuits of this nature can be based on either the generator polynomial, $\boldsymbol{g}(X)$, or parity polynomial, $\boldsymbol{h}(X)$, of a code [27]. Attention is initially focused on architectures derived from the generator polynomial.

Encoding of cyclic codes based on the generator polynomial is normally achieved in one of two ways. The underlying principle common to both methods arises from

the fact that a valid codeword must be a multiple of $\boldsymbol{g}(X)$ [28]. Consequently codewords may be generated in either non-systematic form, by directly multiplying the information polynomial by $\boldsymbol{g}(X)$, or in systematic form, by placing the information symbols in either the highest or lowest order positions of the codeword vector and generating the corresponding parity check symbols by suitable polynomial division. The most straightforward of these two methods is the generation of codewords in non-systematic form which, as stated, involves the simple multiplication or convolution of the information polynomial by the generator polynomial $\boldsymbol{g}(X)$.

Figure 3.1 illustrates a circuit for multiplying an arbitrary information polynomial, $\boldsymbol{u}(X)$, by the generator polynomial $\boldsymbol{g}(X) = 1 + g_1X + ... + g_{n-k-1}X^{n-k-1} + X^{n-k}$, where for simplicity the coefficients of $\boldsymbol{u}(X)$ and $\boldsymbol{g}(X)$ are defined over GF(2) i.e. a binary code. The information bits enter the circuit from the left hand side with the most significant bit entering first, it is assumed that the shift registers initially contain zeros. After n successive clock cycles a codeword corresponding to the information vector will have been generated on a bit-by-bit basis.

The linear nature and operation of this autonomous circuit results in a one to one mapping between an information vector and a codeword. This approach to codeword generation allows all 2^k binary codewords to be produced in an efficient manner without the need for large look up tables.

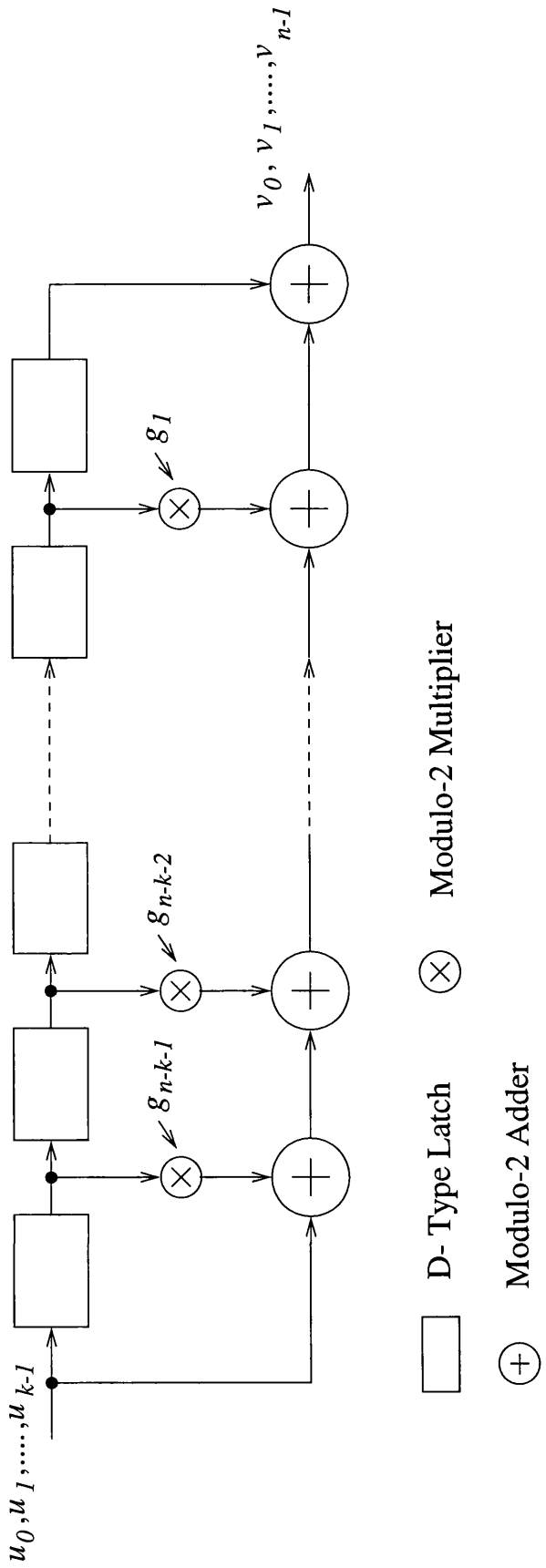


Figure 3.1: Multiply by $g(X)$ circuit.

To clarify the operation of such a circuit a simple example based on the (7,4) Hamming code with generator polynomial $\boldsymbol{g}(X) = 1 + X + X^3$ is considered. Suppose the information vector to be encoded is $\underline{\mathbf{u}}$=0111, or represented in polynomial form, $\boldsymbol{u}(X) = X + X^2 + X^3$. The corresponding codeword is then given by the modulo-2 multiplication of $\boldsymbol{u}(X)$ and $\boldsymbol{g}(X)$ i.e.

Message $\boldsymbol{u}(X)$	Encoding	Codeword $\boldsymbol{v}(X)$
0 1 1 1	$\boldsymbol{u}(X).\boldsymbol{g}(X)$	0 1 0 0 0 1 1
$X + X^2 + X^3$	$(X + X^2 + X^3).(1 + X + X^3)$	$X + X^5 + X^6$

Where . denotes multiplication.

The circuit of figure 3.1 now translates into that of figure 3.2, where the modulo-2 multipliers are realized by a short circuit for multiply by 1 and an open circuit for multiply by 0.

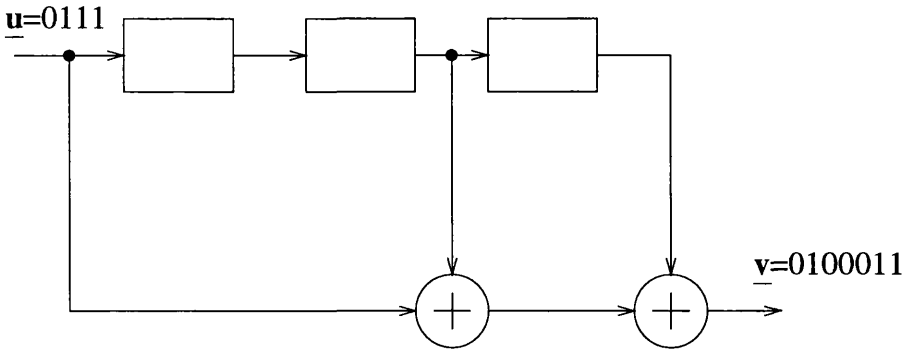


Figure 3.2: Multiply by $X^3 + X + 1$ circuit.

Although a valid method for producing codewords, circuits of this nature are seldom used. As may be observed from this example the codeword does not contain the information vector in its original form. Instead the information has been embedded in the codeword by the encoding process. In order to recover the original message the decoder must now divide the received message by $\boldsymbol{g}(X)$ once any error detection/correction has been performed. To overcome the need for additional circuitry

it is common practice to generate codewords in systematic form using polynomial division circuits. In chapter 2 a systematic codeword was shown to contain the unaltered information vector in the high order positions and the parity check symbols in the low order positions. Consequently, a decoder may now be constructed which recovers the original message by simply discarding the parity check symbols following any error detection/correction. For an (n, k) cyclic code, systematic encoding may be achieved as follows.

1. Premultiply the message, $\boldsymbol{u}(X)$, by X^{n-k}.
2. Obtain the remainder, or parity check symbols, by dividing $\boldsymbol{u}(X).X^{n-k}$ by the generator polynomial $\boldsymbol{g}(X)$.
3. Combine the parity check symbols with the information symbols to form a codeword.

Figure 3.3 illustrates a commercially available encoder, based on polynomial division techniques, for generating an (n, k) cyclic code with symbols from GF(2^m) [29]. In order to generate a codeword the circuit performs the three previously described steps as follows: Step 1, pre-multiplication by X^{n-k} is implicit by the configuration of the circuit. Note that with the gate closed the message enters the circuit from the right hand side as opposed to the left hand side of figures 3.1 and 3.2. Step 2, calculation of parity check symbols, starts with the information vector simultaneously entering the transmission channel and circuit through the closed gate. After all k symbols have entered the parity check symbols are contained within the $(n-k)$ storage elements. Finally, step 3 is achieved by clocking the contents of the storage elements into the transmission channel with the feedback gate open to realise a contiguous n symbol codeword. The storage elements are then reset to zero and the process repeated for subsequent information vectors.

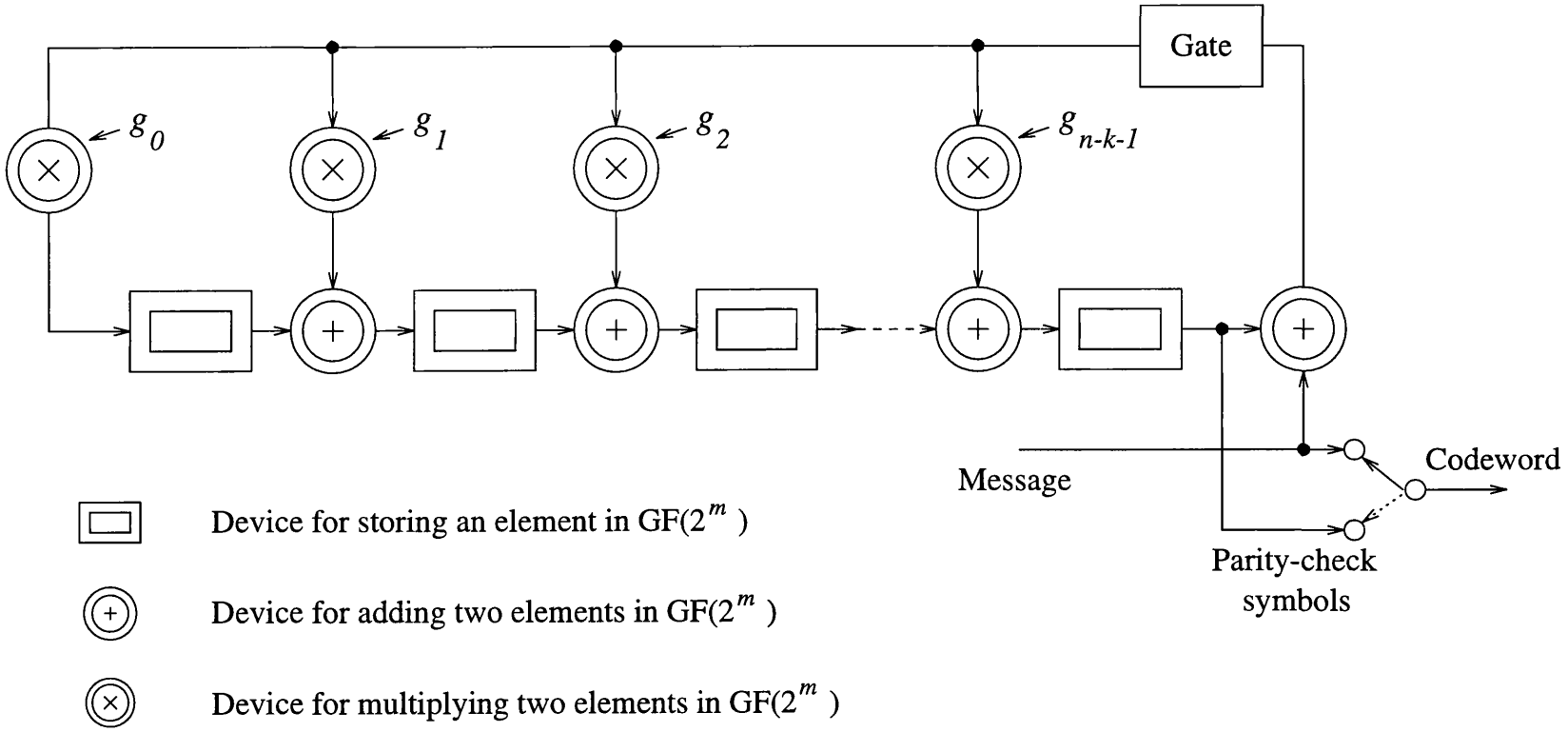


Figure 3.3: Polynomial division encoder based on generator polynomial.

Returning to the previous example of the (7,4) Hamming code, the following circuit may be constructed.

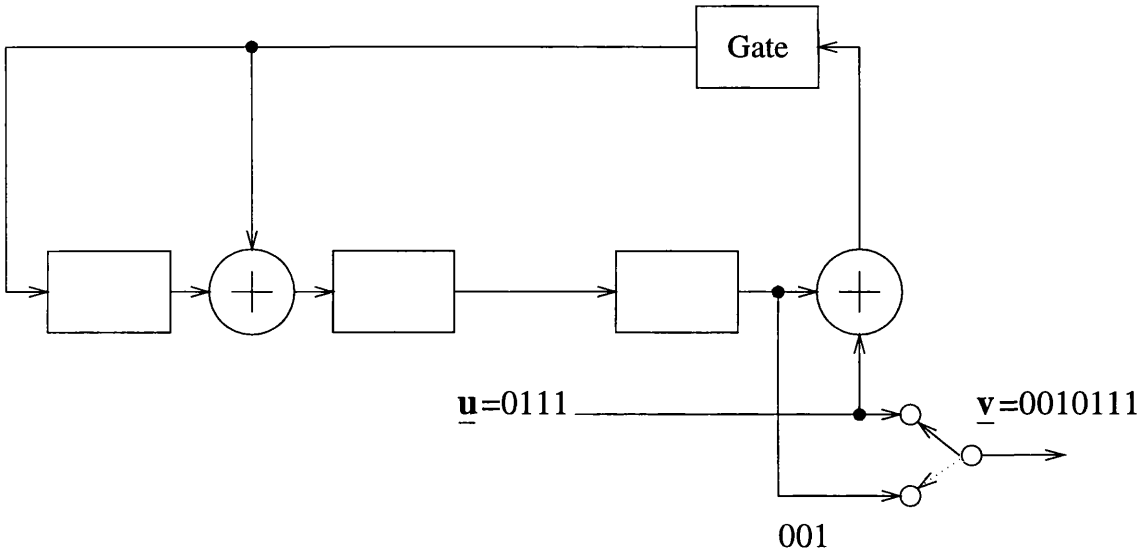


Figure 3.4: Systematic (7,4) Hamming encoder based on the generator polynomial $g(X) = 1 + X + X^3$.

Again, using the information vector $\underline{u}$=0111 as an example the following table illustrates the shift register contents for each successive clock cycle. With the initial

Input	Register Contents
	0 0 0 Initial state
1	1 1 0 1st shift
1	1 0 1 2nd shift
1	0 1 0 3rd shift
0	0 0 1 4th shift

Table 3.1: Shift register contents.

contents of the registers set to zero it may be observed that during the first 4 clock cycles the information enters both the circuit and transmission channel. On comple-

tion of the fourth shift the parity check digits have been calculated and are stored within the three registers. For the final 3 clock cycles the switch is placed in the down position and the parity check digits 001 are clocked out of the circuit and combined with the information vector to form the complete codeword $\underline{\mathbf{v}}$=0010011.

Encoding of a cyclic code can also be accomplished by using its parity polynomial. Figure 3.5 shows the general form of a systematic encoder based on the parity polynomial $\boldsymbol{h}(X) = h_0 + h_1X + h_2X^2 + ... + h_{k-1}X^{k-1} + X^k$.

The function of this circuit is to realise the recursive or difference equation (3.1):

$$v_{n-k-j} = \sum_{i=0}^{k-1} h_i v_{n-i-j} \qquad 1 \leq j \leq n-k \tag{3.1}$$

i.e.

$$\begin{aligned} v_{n-k-1} &= h_0v_{n-1} + h_1v_{n-2} + \ldots + h_{k-1}v_{n-k} \\ &= u_{k-1} + h_1u_{k-2} + \ldots + h_{k-1}u_0 \qquad (3.2) \\ v_{n-k-2} &= h_0v_{n-2} + h_1v_{n-3} + \ldots + h_{k-1}v_{n-k-1} \\ &= u_{k-2} + h_1u_{k-3} + \ldots + h_{k-2}u_0 + h_{k-1}v_{n-k-1} \qquad (3.3) \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ v_0 &= v_k + h_1v_{k-1} + \ldots + h_{k-1}v_1 \qquad (3.4) \end{aligned}$$

The functionality of this circuit is described as follows. With Gate 1 closed and Gate 2 open, the information vector is simultaneously clocked into the circuit and transmission channel. When all k information symbols have entered, Gate 1 is opened and Gate 2 closed. Each subsequent clock cycle then represents a single summation of equation (3.1) i.e. the circuit systematically computes v_{n-k-j} for $1 \leq j \leq n-k$. The $(n-k)$ parity symbols are then clocked out of the circuit to form the codeword. As with generator polynomial based encoders, the storage elements are then reset to zero ready for the following information vectors.

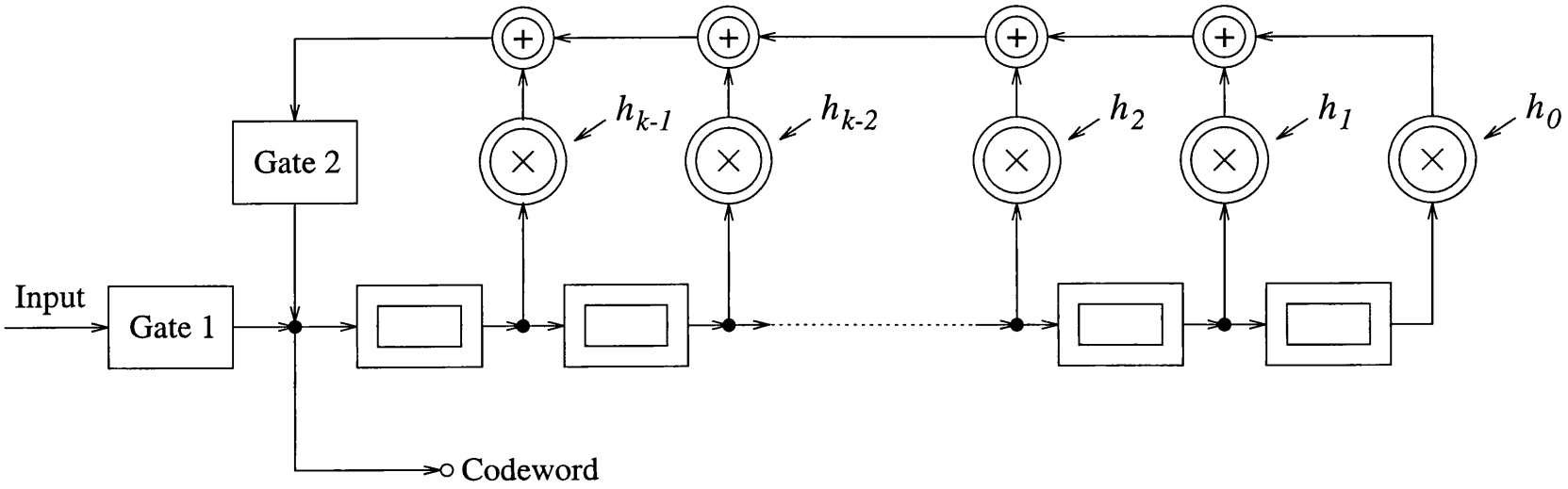


Figure 3.5: Systematic encoder based on parity polynomial.

In the case of the (7,4) Hamming code defined by the parity polynomial $\boldsymbol{h}(X) = 1+ X + X^2 + X^4$ the encoder shown in figure 3.6 can be constructed.

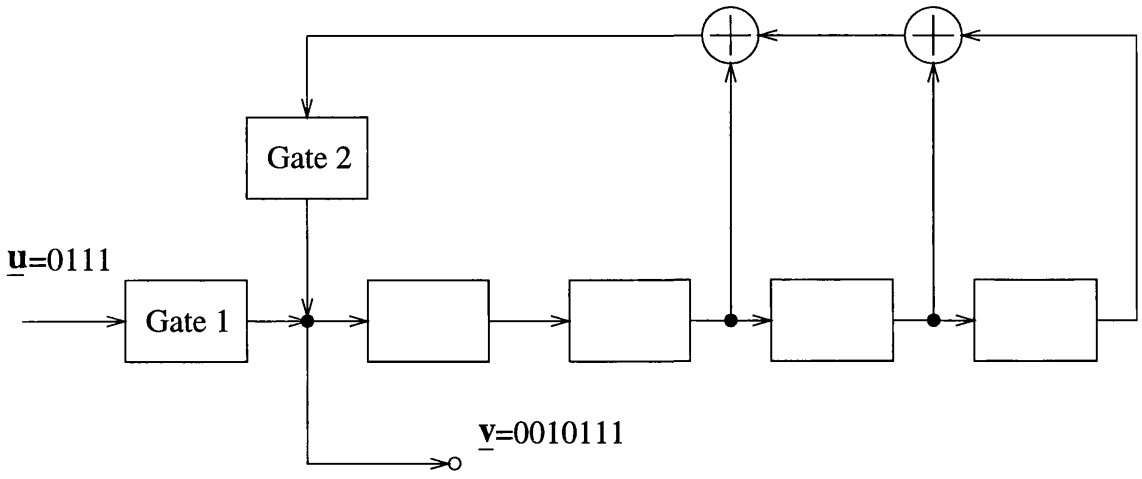


Figure 3.6: (7,4) Hamming encoder based on parity polynomial.

	Input	Register Contents
Information Vector		0 0 0 0 Initial state
	1	1 0 0 0 1st shift
	1	1 1 0 0 2nd shift
	1	1 1 1 0 3rd shift
	0	0 1 1 1 4th shift
	Output	
Parity Check Bits	1	1 0 1 1 5th shift
	0	0 1 0 1 6th shift
	0	0 0 1 0 7th shift

Table 3.2: Contents of shift registers.

By examining the contents of each register, as shown in table 3.2, it may be observed that the initial parity check digit is formed after the fourth cyclic shift of the LFSR network. The circuit then yields a further parity check digit on each successive shift

of the registers until all $(n-k)$ digits have been generated.

Although the previous examples relate only to binary codes the operation and construction of such circuits can be easily extended to multi-level codes. Having demonstrated the operation of several encoding circuits it is now appropriate to consider the various performance and complexity issues relating to each arrangement.

Code generation by polynomial multiplication, while offering a more direct approach, often requires a more complex decoding arrangement. Producing codewords in systematic form, by either polynomial division or encoders based on a code's parity polynomial, effectively eliminates this need for additional decoding circuitry. Although the two systematic encoders produce identical codes there exist a number of subtle differences between them.

In hardware terms, generating codes by polynomial division requires a circuit containing $(n-k)$ storage elements with the check symbols being formed after the k^{th} cyclic shift. In contrast, generating codes via the parity polynomial involves a circuit containing k storage elements which is clocked $(n-k)$ times, with each successive shift producing one additional parity symbol.

In order to minimise circuit complexity most commercially available encoders are based on architectures derived from the generator polynomial. Consider, for example, the RS (255,239) code. Generating this code would require a circuit consisting of either 239 storage elements for a parity polynomial based encoder or 16 storage elements for a generator polynomial based encoder.

Having identified which circuit architecture is more efficient in terms of hardware, the factors determining the maximum operational speed of each encoding arrangement are now considered. Each encoder must be clocked a total of n times in order to generate a complete codeword. With every clock cycle resulting in the transmission

of a single codeword symbol, the data throughput of each arrangement is determined by the maximum clock frequency that can be applied to each circuit.

It is clear from figures 3.3 and 3.5 that the minimum clock period, and hence the maximum clock frequency, is dependent on several timing parameters. These parameters include: the propagation delays due to the addition and multiplication devices plus the propagation delay, minimum set-up time and minimum hold time associated with the storage elements. Assuming all these factors are constant for both the parity and generator polynomial based circuits, the minimum clock period is determined by the longest delay encountered by a codeword symbol.

By observation of figure 3.3 a codeword/information symbol will, in any one clock cycle, traverse through one multiplication device, a storage element and at most two addition devices. In contrast, an encoder based on the parity polynomial (see figure 3.5) will have an equal delay due to a single multiplication device and storage element, but will have an increased delay due to the $(n_h - 2)$ addition elements, where n_h is the number of non-zero coefficients in the parity polynomial. Generally this value is much greater than the delay due to the two addition devices found in generator polynomial based circuits. As a result of the increased delays imposed by such an arrangement, encoders based on the parity polynomial are rarely used in high speed applications.

In summary, it has been demonstrated that architectures based on the generator polynomial of a cyclic code can generally provide a solution which is both faster and less complex than that of parity polynomial based circuits.

3.2.2 Parallel Encoding Architectures

In the previous section encoders based on serial architectures were introduced. Although convenient to implement, the necessary cyclic shifting of the LFSR network required to generate codewords made such arrangements somewhat slow. In order to overcome these speed limitations it is possible to generate the codewords, or the parity check symbols, directly from the parity check equations. We may recall from the previous section that the expansion of equation (3.1) produced $(n-k)$ linearly independent equations. While a serial encoder based on the parity polynomial calculates each of the parity check digits sequentially it is possible to construct an encoder which will calculate all the parity check digits simultaneously.

For an (n,k) cyclic code in systematic form, the information digits $u_0, u_1, ..., u_{k-1}$ form the components $v_{n-k}, v_{n-k+1}, ..., v_{n-1}$ of the code vector. Given the k information symbols, equations [3.2 - 3.4] define the $(n-k)$ parity check symbols $v_0, v_1, ..., v_{n-k-1}$. For the (7,4) hamming code defined by the parity polynomial $\boldsymbol{h}(X) = 1 + X + X^2 + X^4$ the resulting parity check equations are :

$$\begin{aligned} v_2 &= v_6 + v_5 + v_4 \\ &= u_3 + u_2 + u_1 \end{aligned} \tag{3.5}$$

$$\begin{aligned} v_1 &= v_5 + v_4 + v_3 \\ &= u_2 + u_1 + u_0 \end{aligned} \tag{3.6}$$

$$\begin{aligned} v_0 &= v_4 + v_3 + v_2 \\ &= u_3 + u_2 + u_0 \end{aligned} \tag{3.7}$$

Figure 3.7 illustrates an encoder based on the three parity check equations of the previous example. As may be observed, the information vector is now presented to the encoder in a parallel form. It is this inherent parallelism which provides the

basis for increased operational speed. Since all parity check digits are calculated from the already present information bits no feedback path is required. As a result the operational speed of such a circuit is determined by the asynchronous delay of the EXOR-gate network.

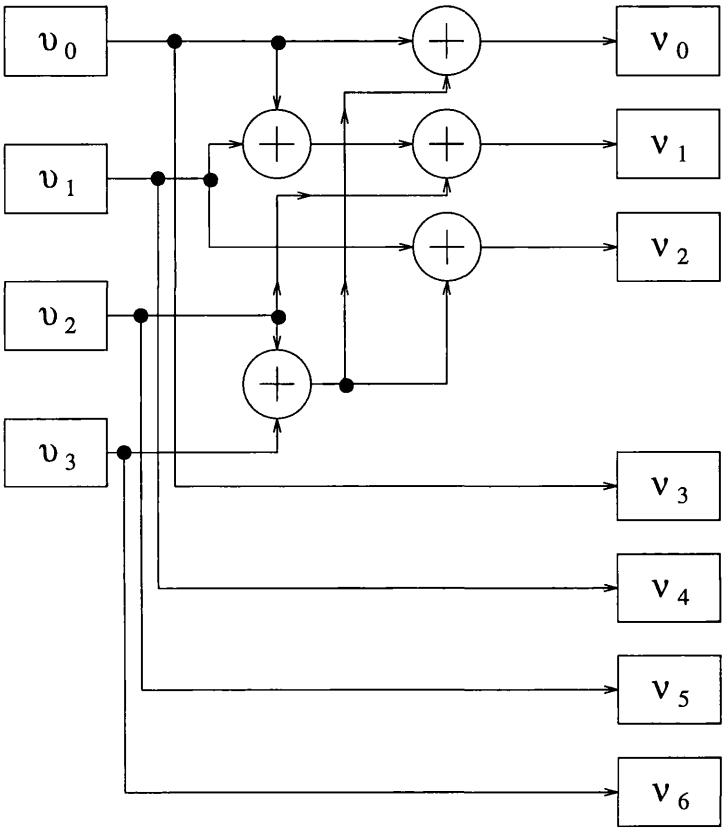


Figure 3.7: Parallel (7,4) Hamming Encoder.

However, while offering the prospect of a potentially fast solution, circuits of this nature are generally too complex for all but the most trivial of codes. To demonstrate this, figure 3.8 illustrates a parallel encoder capable of generating a (15,11) single error correcting code, based on the parity polynomial $\boldsymbol{h}(X) = 1 + X + X^2 + X^3 + X^5 + X^7 + X^8 + X^{11}$.

Using the parity polynomial and equation (3.1) it is once again possible to define the following parity check digits.

$$v_3 = u_{10} + u_9 + u_8 + u_7 + u_5 + u_3 + u_2 \tag{3.8}$$

$$v_2 = u_9 + u_8 + u_7 + u_6 + u_4 + u_2 + u_1 \tag{3.9}$$

$$v_1 = u_8 + u_7 + u_6 + u_5 + u_3 + u_1 + u_0 \tag{3.10}$$

$$v_0 = u_{10} + u_9 + u_8 + u_6 + u_4 + u_3 + u_0 \tag{3.11}$$

By comparing the encoder of figure 3.8 with that of figure 3.7 it is clear that the overall circuit complexity has increased substantially. In this example a four fold increase in the number of EXOR-gates is the result of approximately doubling the codeword length. This problem is further compounded at larger block lengths and therefore limits the effectiveness of this approach to encoding.

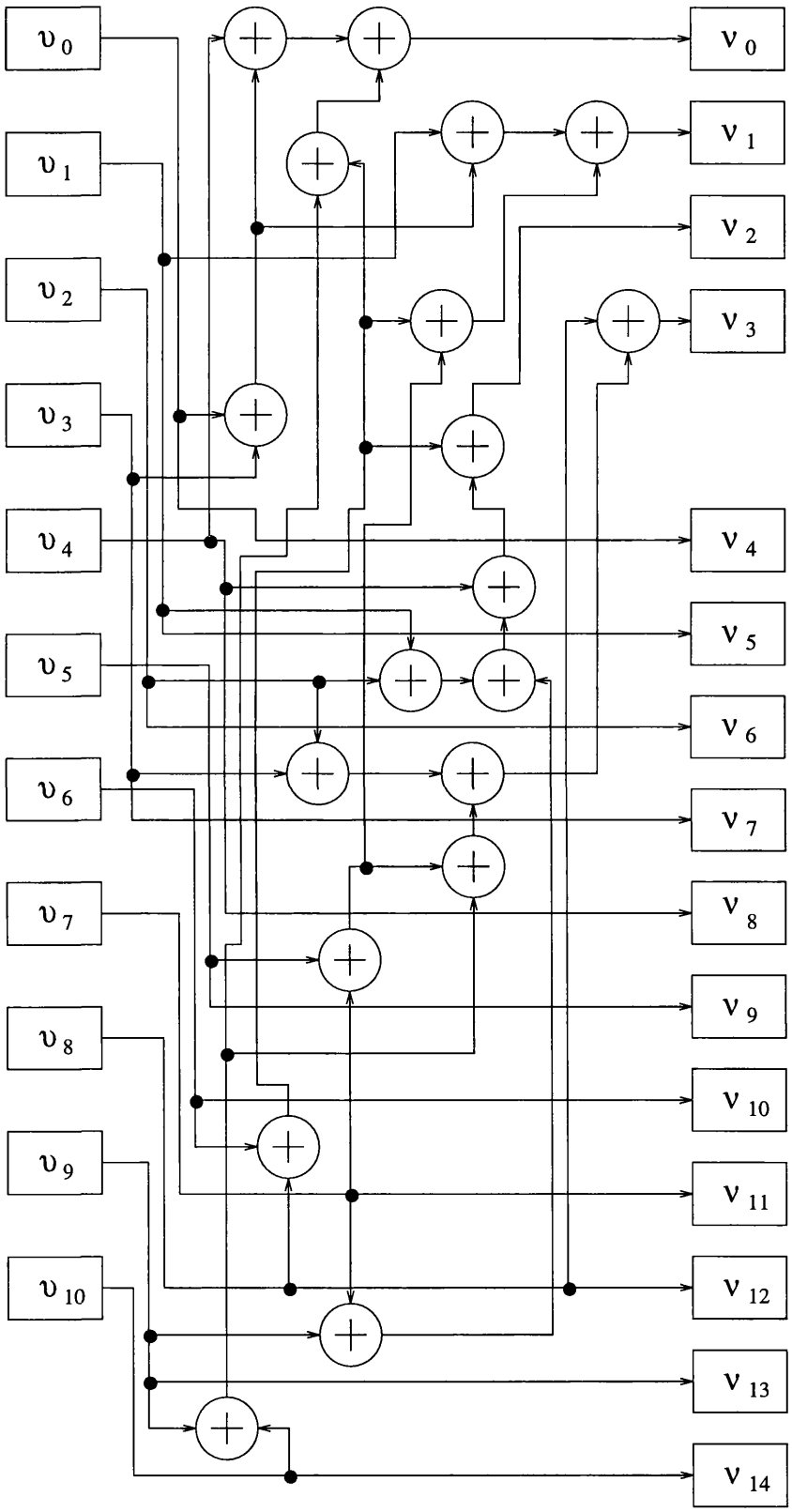


Figure 3.8: Parallel (15,11) Hamming Encoder.

Having presented various encoding strategies in the previous sections attention is now directed towards error detection or syndrome calculation circuits based on similar architectures.

3.3 Error Detection Architectures

In a manner resembling the encoding process syndrome calculation may be achieved by a network of LFSRs configured to perform polynomial division. Here only binary structures are considered since multi-level error detection circuits prove to be trivial extensions. Figure 3.9 illustrates a binary polynomial division circuit which calculates the remainder resulting from the division of $\boldsymbol{r}(X) = r_0 + r_1X + ... + r_{n-1}X^{n-1}$ by the minimal polynomial $\boldsymbol{\Phi}_i(X) = \phi_{i,0} + \phi_{i,1}X + \phi_{i,2}X^2 + ... + \phi_{i,m-1}X^{m-1}$, where $\boldsymbol{\Phi}_i(X)$ has α^i as a root. This, however, only represents part of the error detection process. To enable S_i to be calculated additional logic must be incorporated in order to evaluate the remainder at α^i. Since this logic consists merely of addition elements and operates only on the output of the circuit it does not impede the operational speed and can therefore be ignored in any further timing analysis. For a t error correcting code of length $n = 2^m - 1$, t such circuits each containing m storage elements are required in order to calculate the syndromes $S_1, S_3, ..., S_{2t-1}$.

As figure 3.9 demonstrates, the syndrome calculation circuit is similar to that of the encoding circuit based on the generator polynomial. The only difference being the received vector enters the circuit from the left hand side. Consequently the operational speed of this circuit is also governed by the longest path encountered by a received symbol, which in this example is an addition element, a multiplier and a storage device. However, in this instance the delay due to the multiplier may be ignored as it would be replace by either an open or short circuit in any practical implementation.

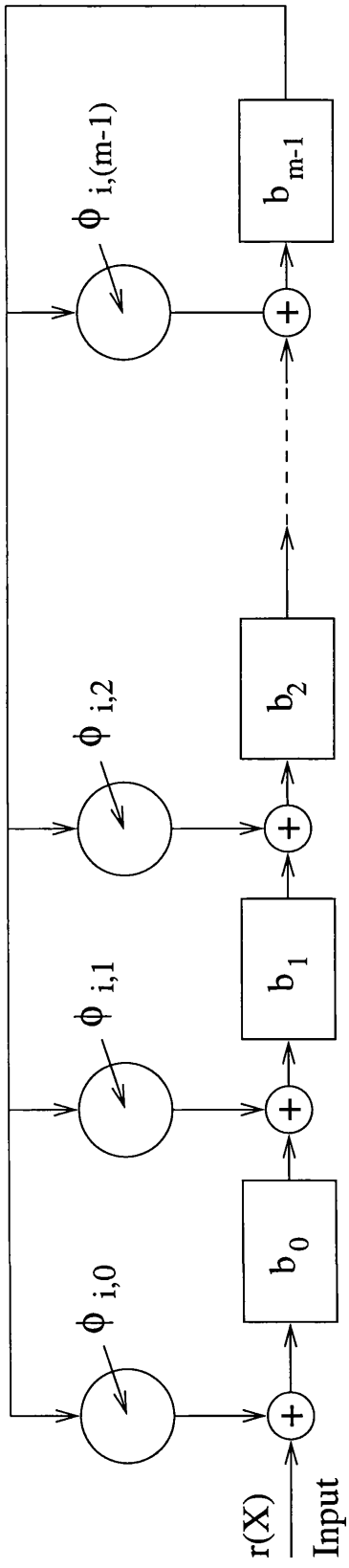


Figure 3.9: Polynomial division based on the minimal polynomial $\mathbf{\Phi}_i(X) = \phi_{i,0} + \phi_{i,1}X + \phi_{i,2}X^2 + \ldots + \phi_{i,m-1}X^{m-1}$.

After each of the t syndrome circuits has been clocked a total of n times the syndrome components $S_i = \{s_{i,0}, s_{i,1}, ..., s_{i,m-1}\}$ are held within the m storage elements. Although convenient to implement, this solution is again restricted by the cyclic shifting required to generate syndrome elements. With this in mind parallel syndrome circuits are now considered.

To implement error detection in a parallel form the syndrome components based on the parity check equations are re-defined. This simply involves re-calculating the parity check digits from the received vector and comparing them to the actual values. To illustrate this the syndrome vector, $S_1 = \{s_{1,0}, s_{1,1}, s_{1,2}\}$, for the (7,4) Hamming code is defined as follows:

$$s_{1,0} = r_0 + r_3 + r_5 + r_6 \tag{3.12}$$

$$s_{1,1} = r_1 + r_3 + r_4 + r_5 \tag{3.13}$$

$$s_{1,2} = r_2 + r_4 + r_5 + r_6 \tag{3.14}$$

If no errors have occurred then $r_i = v_i$ for $0 \leq i \leq n-1$ and the syndrome vector is found to be zero. These equations can now be used to define the following parallel error detection circuit.

As figure 3.10 illustrates the parallel syndrome calculation circuits have a similar form to that of the parallel encoding circuits. Although accommodating high speed operation it is again evident that circuits of this nature are restricted, by constraints placed on the circuit complexity, to simple codes with short block lengths.

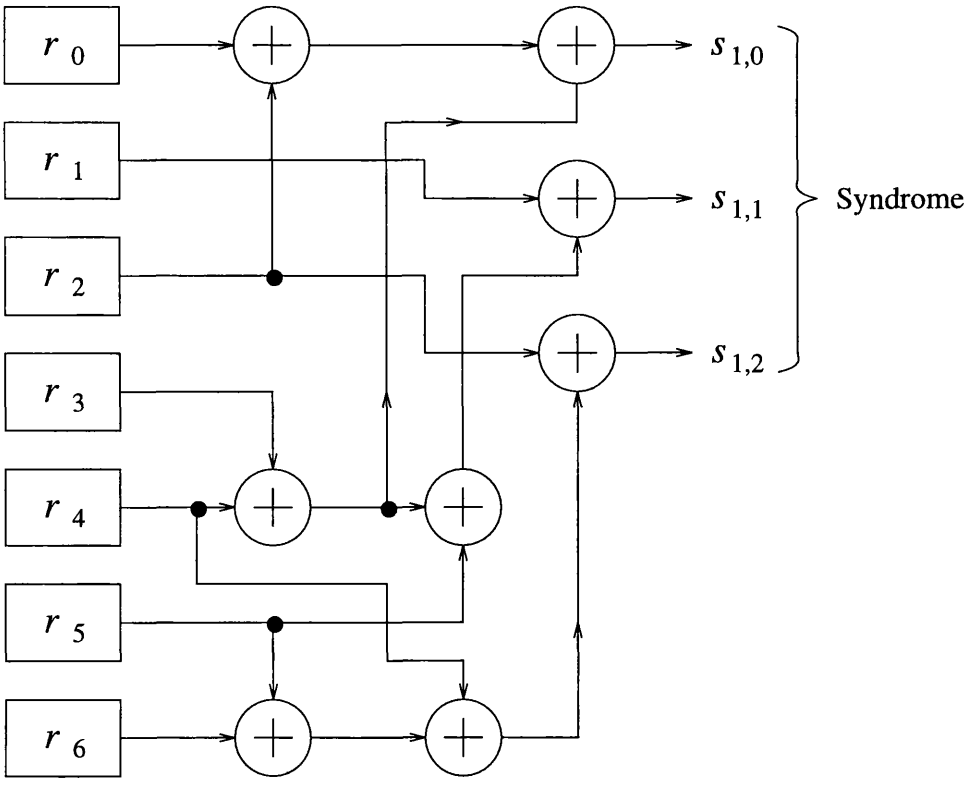


Figure 3.10: Parallel syndrome calculation circuit.

3.4 Summary

In this chapter standard architectures for the realisation and error detection of BCH codes have been described. In the case of serial arrangements it was demonstrated that the necessary cyclic shifting of the LFSR networks rendered such arrangements unsuitable for high speed applications. An alternative approach, which overcomes these limitations, was presented in the form of parallel circuits. Although offering the prospect of increased operational speed it was found that circuit complexity increases rapidly with codeword length. In the following chapter new architectures will be developed and investigated which will allow high speed encoding and error detection to be performed in an efficient manner.

Chapter 4

Series-Parallel Encoding and Error Detection

4.1 Introduction

Chapter 3 introduced standard architectures employed for the realisation of encoding and error detection systems. It was demonstrated that such architectures were often too slow or too complex to be of any value in high speed applications. Consequently, new techniques devised for encoding and error detection at very high bit rates are now investigated.

Specifically these make use of what are termed series-parallel arrangements which offer the prospect of a trade off between operational speed and circuit complexity, the degree of parallelism adopted influencing the achievable coded line rate relating to the overall system clock rate.

Using m-sequence generation architectures to introduce series-parallel concepts, these techniques are then adapted to encoding arrangements where previous results are extended to encompass non-binary as well as binary coding arrangements. We then examine how the series-parallel circuit techniques may be applied to syndrome calculation or error detection circuits. The speed/complexity trade off afforded by these arrangements are then quantified and summarised graphically.

4.2 Series-Parallel Architectures

In previous sections two distinct approaches to encoding and error detection have been clearly defined. The serial approach where the speed of operation is compromised in favour of reduced circuit complexity and the parallel approach where an increase in speed is obtained at the expense of a prohibitively large increase in circuit complexity. In an attempt to find a solution which offers a compromise between these two extremes we turn our attention to series-parallel architectures.

4.2.1 m-sequence Generation

In order to provide a background for series-parallel architectures we make recourse to previous work concerning m-sequence generation. Pseudo-random bit sequences are traditionally generated in a serial manner by a network of linearly interconnected feedback shift registers. The configuration of such a network is chosen so as to produce an m-sequence of maximal length $2^m - 1$, where m is the number of shift registers. However, generating m-sequences in such a manner is often restrictive when operating at high speeds. Previously reported solutions to this problem have relied on several pairs of shift registers operating in parallel with their outputs

phase shifted [30]. Here the number of replicated shift register networks employed determines the overall bit rate. Although convenient to implement at relatively modest speeds, this approach becomes unduly complex when operating at high bit rates (e.g. $>$ 500 Mb/s), especially when long sequences are required.

This led O'Reilly [31] to develop architectures based on series-parallel concepts. Here the m-sequences are generated as sets of non-overlapping k-bit words and unlike the previous approach this method achieves high speed operation using only a single LFSR network.

To effect a speed increase the serial LFSR network is mapped into an equivalent series-parallel form by way of a matrix description of the circuit. The transition matrix defines the functional specification of circuit and allows the subsequent state of the registers to be derived from their present state.

For a LFSR of length m we define U(j) as a $(m \times 1)$ column vector which describes the contents of the m registers after the j^{th} clock cycle and T, the transition matrix, as an $(m \times m)$ matrix with elements defined over the binary field GF(2). If each stage of the LFSR is numbered sequentially, as shown in figure 4.1, then each row of the transition matrix defines the excitation of that stage in the network. For example the circuit in figure 4.1, as described by the following transition matrix, generates an m-sequence of maximal length 31 with the output sequence being derived from register five.

$$T = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.1}$$

From this matrix we see that row 1 implies that stage 1 is derived from the modulo-2 addition of stages 3 and 5, row 2 implies that stage 2 is derived from stage 1 and so on.

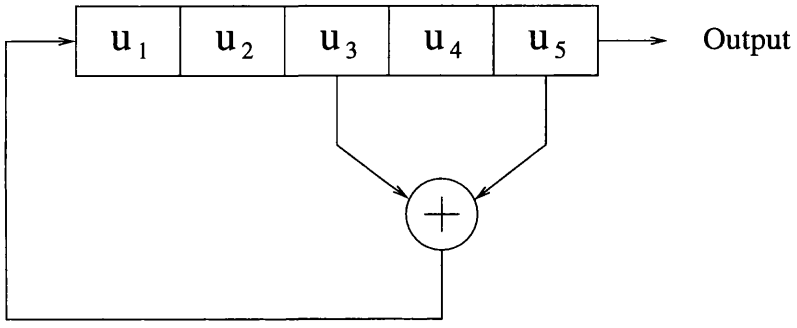


Figure 4.1: Serial m-sequence generator.

The state of the LFSR resulting from each successive clock cycle is defined as the modulo-2 multiplication of U(j), the present register contents, and T thus

$$U(j+1) = T.U(j) \tag{4.2}$$

$$\begin{aligned} U(j+2) &= T.U(j+1) \\ &= T^2.U(j) \end{aligned} \tag{4.3}$$

$$\vdots \qquad \vdots$$

$$U(j+k) = T^k.U(j) \tag{4.4}$$

For example, if the current contents of the registers is (1 0 0 1 1) then the following state may be calculated as follows.

$$U(j+1) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.5}$$

similarly

$$U(j+2) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.6}$$

Alternatively U(j+2) may be calculated from the initial contents using T^2 thus.

$$U(j+2) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.7}$$

Equation (4.4) and the previous examples demonstrate that the contents or state of the LFSR after k clock cycles can be obtained in a single clock cycle using an equivalent circuit defined by T^k. To demonstrate this, the following illustrative examples based on T^2 and T^3 are given.

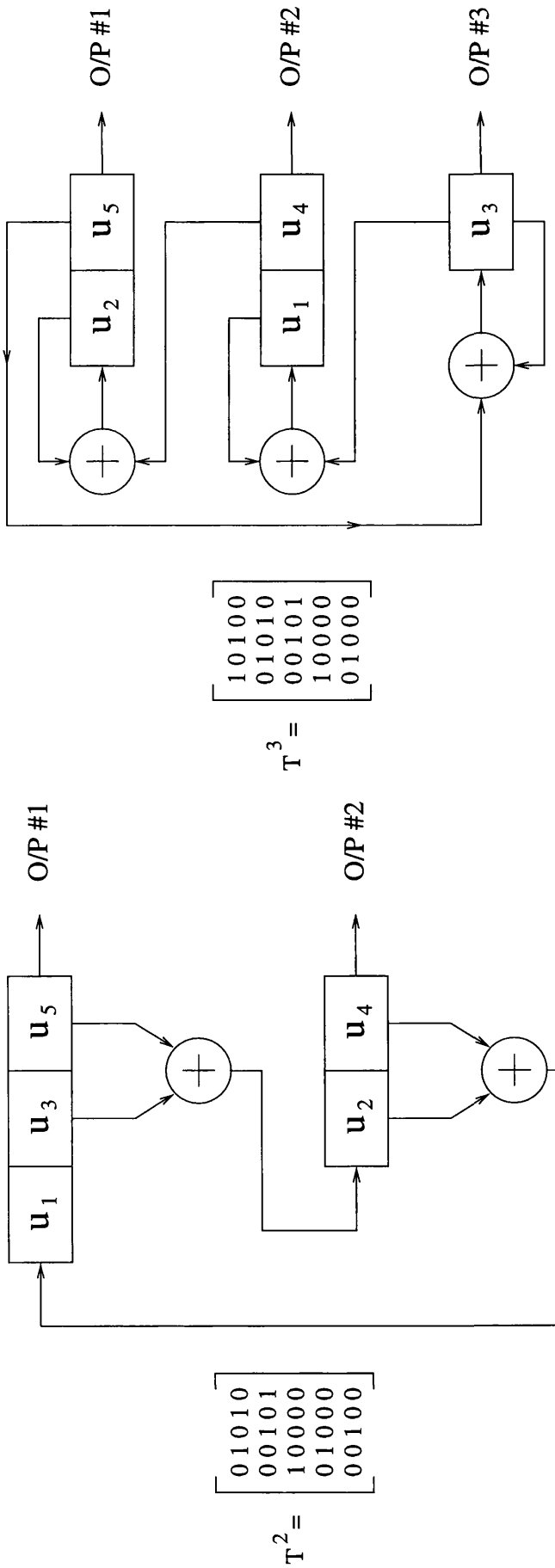


Figure 4.2: Illustrative examples of series-parallel m-sequence generation based on T^2 and T^3.

By raising T to the power of 2, as shown in figure 4.2a, a circuit which generates two output bits per clock cycle has now been produced. This two fold increase in speed is obtained at the expense of a small increase in circuitry. As expected the number of storage elements remains constant but the number of EXOR-gates has increased. Extending this process further to the case of T^3, as shown in figure 4.2b, the introduction of an additional EXOR-gate allows a three fold increase in speed over the serial implementation.

To conclude this illustration, a fully parallel implementation of the circuit may be defined by calculating T^5 as shown in matrix (4.8).

$$T^5 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{4.8}$$

The resulting circuit shown in figure 4.3, although the most complex of those considered, produces the same output sequence as that of figure 4.1 but now generates five consecutive bits per clock cycle. By serialising the contents of the shift registers a bit rate of 5 times that of the serial implementation may be realised. Since several consecutive bits of the m-sequence are produced simultaneously the structure of the sequence is preserved.

By using this approach the high speed emphasis has been removed from the LFSR network and placed on the multiplexing operation. With current multiplexers or parallel to serial converters operating in the Gbit/s region [32] [33] the generation of m-sequences at high bit rates may be achieved with conventional logic families.

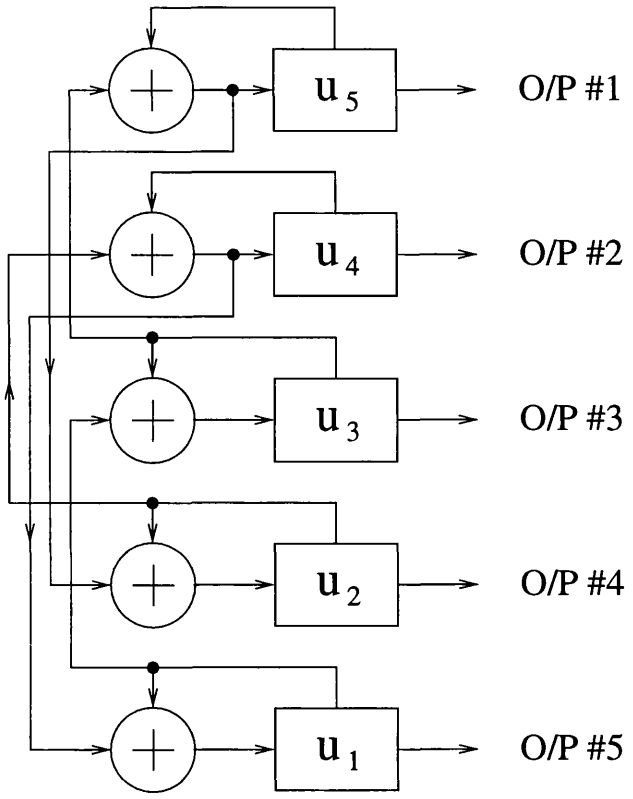


Figure 4.3: Parallel m-sequence generator.

We can therefore conclude that by raising T to various powers we are able to generate equivalent circuits with varying degrees of parallelism, thus enabling the rate at which the sequences are generated to be increased with only a modest increase in circuit complexity. Moreover, intermediate circuits have been identified which enable a trade off between the degree of parallelism incorporated and circuit complexity.

4.2.2 Series-Parallel Code Generation

Having introduced the general theory and concepts of series-parallel m-sequence generation we now adapt and apply these ideas to the process of encoding cyclic codes. We may recall that encoding may be achieved by generating the check bits using a LFSR network. Since the use of LFSR networks for code generation essentially involves a similar procedure to that of m-sequences, it is appropriate to extend

series-parallel theory to the calculation of the parity check digits.

Two different methods for consideration are: (1) Polynomial division circuits based on the generator polynomial and (2) Parity polynomial based circuits. By applying series-parallel techniques to both encoding architectures it is again possible to identify alternative circuit arrangements which provide a trade off between the speed and complexity factors relating to each method. We begin by investigating method (1).

4.2.3 Generator polynomial based circuits.

For (n, k) cyclic code with generator polynomial $\boldsymbol{g}(X) = g_0 + g_1X + ... + g_{n-k}X^{n-k}$ the encoder may be viewed as an autonomous circuit with the input data being presented to the circuit by the first k storage elements, as shown in figure 4.4.

The functionality of this circuit may be described by the following $n \times n$ transition matrix:

$$\mathrm{T} = \begin{bmatrix} 0 & I_{k-1} & 0_{(k-1)\times(n-k)} & \\ \vdots & & & \\ 0 & \cdots & \cdots & 0 \\ g_{n-i} & 0_{(n-k)\times(k-1)} & g_{n-i} & I_{n-k-1} \\ \vdots & & \vdots & \\ g_{n-i} & & g_{n-i} & 0 \end{bmatrix} \tag{4.9}$$

Where i denotes the row number and I_k denotes a $k \times k$ identity matrix.

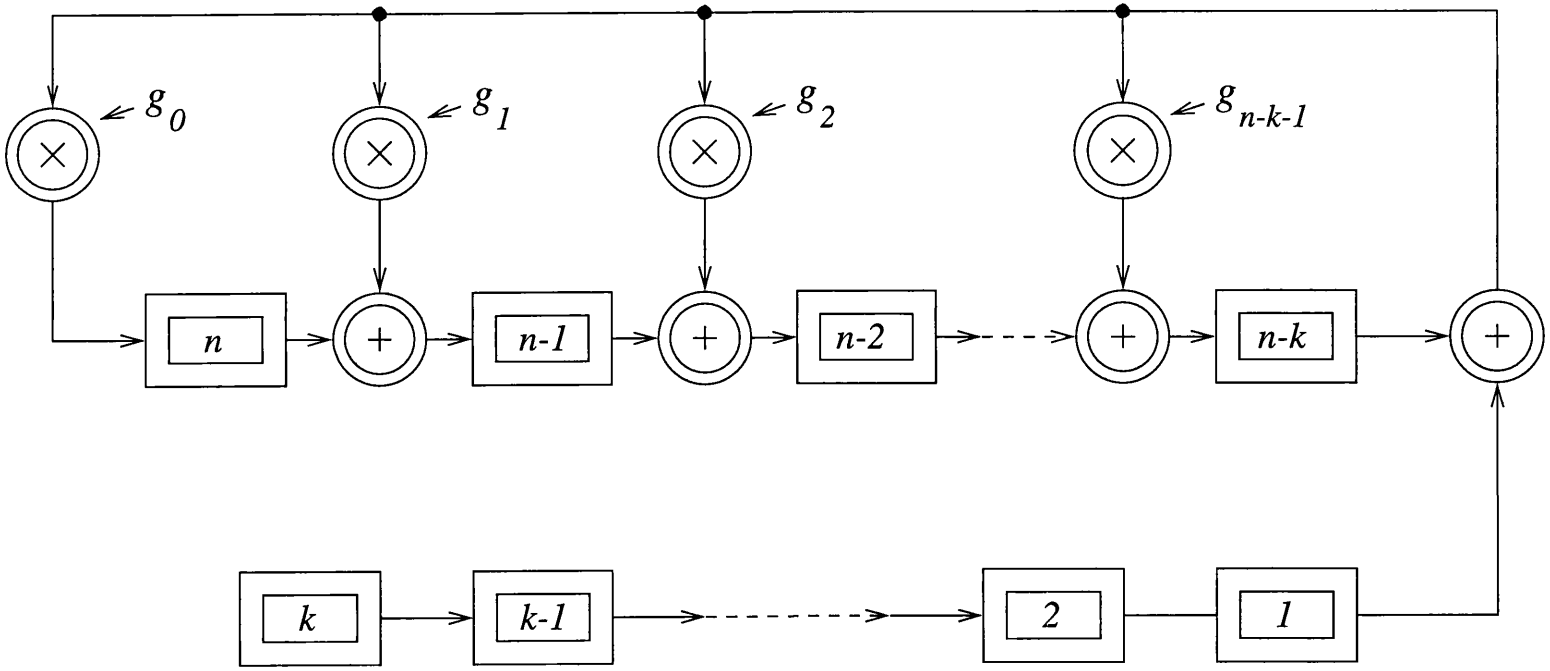


Figure 4.4: Autonomous circuit view of a serial encoder based on the generator polynomial.

For simplicity a binary encoder based on the (7,4) Hamming code is considered. Using the generator polynomial $g(X) = 1 + X + X^3$ as an example, the transition matrix becomes:

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.10}$$

Which defines the serial circuit of figure 4.5.

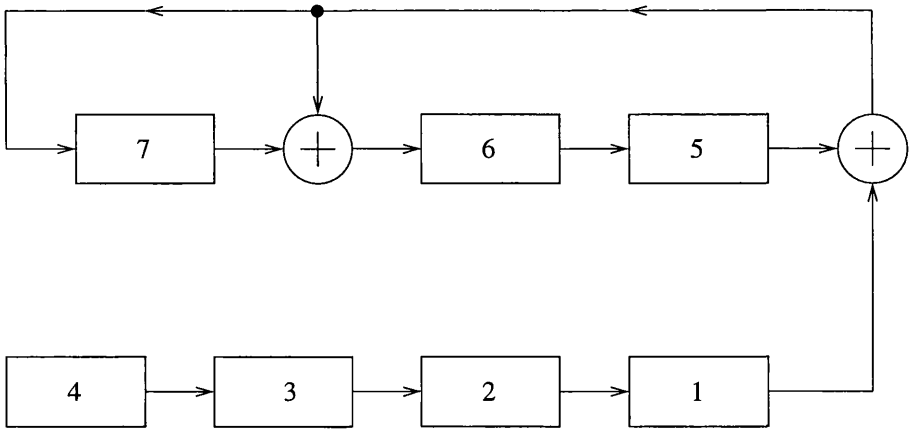


Figure 4.5: Serial Hamming (7,4) encoder.

By raising T to the power of 2 and performing all addition and multiplication over GF(2) it is possible to define a circuit which processes two information bits per clock cycle.

The resulting matrix is:

$$T^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.11}$$

Which defines the following circuit.

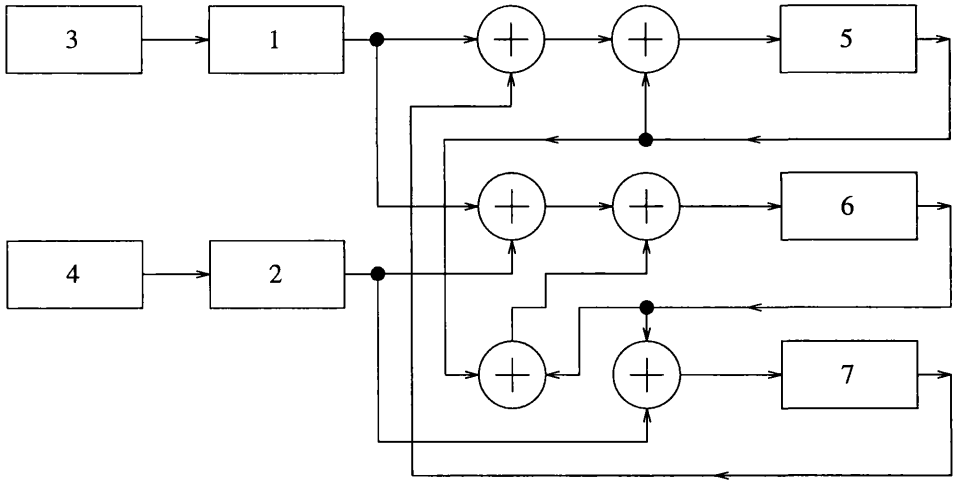


Figure 4.6: Series-parallel (7,4) Hamming encoder.

An intermediate solution has now been identified which, although slightly more complex, calculates the parity check digits after only two clock cycles. It is possible to extend this idea further to obtain a circuit which calculates the required check bits after only one clock cycle. To achieve this T is raised to the power 4, resulting in the following matrix:

$$T^4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{4.12}$$

By calculating T^4 we appear to have produced a circuit which requires six inputs to register 5, 5 inputs to register 6 and five inputs to register 7. Since the circuit is only clocked once and registers 5, 6 and 7 contain zeros at the start of the encoding process these feedback connections may be neglected. As a result the circuit degenerates to that of figure 4.7.

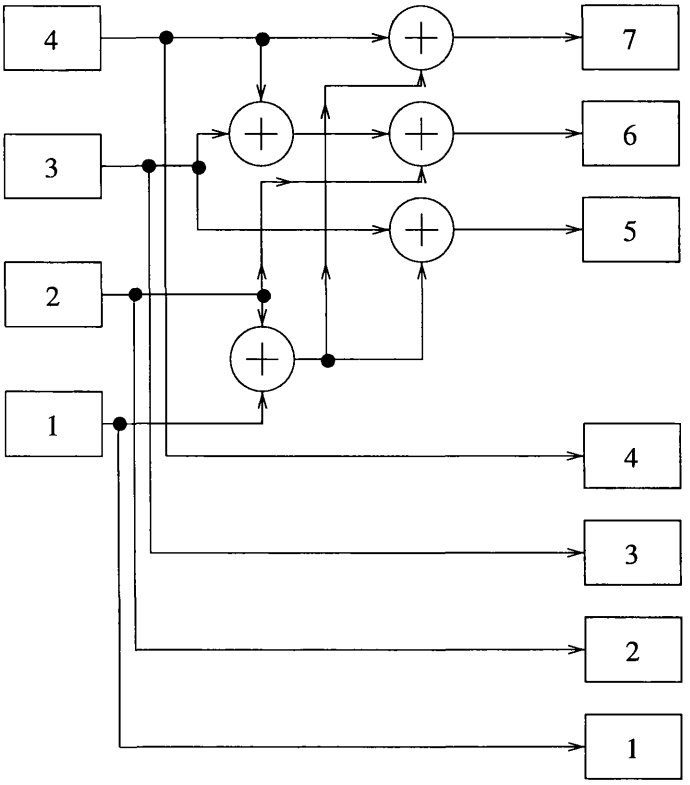


Figure 4.7: Parallel Hamming (7,4) encoder.

By raising T to the power k we arrive at a circuit which calculates the check bits in exactly the same manner as if the parity check equations had been used. In general the degree of parallelism incorporated into the circuit, p, may or may not be a factor of k. In the case where p is a factor of k the the number of clock cycles, s, required to generate the parity check digits is

$$s = \frac{k}{p}$$

However if p is not a factor of k then the circuit must be clocked

$$s = \left\lfloor \frac{k}{p} \right\rfloor$$

times, where $\lfloor x \rfloor$ denotes the largest integer less than x. In this case a shortened code is generated.

Having demonstrated the flexibility of this method to produce solutions with varying degrees of parallelism we now evaluate the practical performance aspects of these circuits in terms of speed and complexity. This is achieved by formulating equations which determine the check-bit computational time and circuit complexity (in terms of equivalent number of gates, where one equivalent gate = one three-input NAND-gate) for various circuit architectures.

Serial polynomial division based circuits:

The general form of these circuits as shown in figure 4.4 consists of $(n-k)$ storage elements (each consisting of m RDT2 resettable D-type flip flops), $(n_g - 1)$ addition elements and $(n_g - 1)$ multipliers, where n_g is the number of non-zero terms in the generator polynomial.

For a BCH code with elements from the Galois Field GF(2^m) the circuit complexity in terms of logic elements, N_{Ts}, is given by

$$\begin{aligned} N_{Ts_{(Non-binary)}} &= m(n-k) \times \text{RDT2s} + (n_g - 1) \times \text{Multipliers} \\ &\quad + (n_g - 1) \times \text{Adders} \end{aligned} \tag{4.13}$$

Throughout this chapter we shall consider an RS (255,k) code where we find that the number of EXOR-gates required to achieve a multiplication function is at most 56 and the number of EXOR-gates required for addition is 8 (see Appendix A). Since each RDT2 occupies six equivalent gates and each EXOR-gate occupies 4 equivalent gates [34]

$$N_{Ts_{(Non-binary)}} = 48(n-k) + 224(n_g - 1) + 32(n_g - 1) \tag{4.14}$$

For binary codes equation (4.14) can be reduced to

$$N_{Ts_{(Binary)}} = (n-k) \times \text{RDT2s} + (n_g - 1) \times \text{EXOR} - \text{gates} \tag{4.15}$$

i.e.

$$N_{Ts_{(Binary)}} = 6(n-k) + 4(n_g - 1) \tag{4.16}$$

Having defined expressions for the complexity of various serial coding arrangements we now turn our attention to the problem of calculating the check-symbol computational time. In order to proceed we must derive an expression for the minimum clock period which can be applied to the circuit. To do this we define the following parameters.

t_{CLKMIN} = minimum clock period that can be applied

t_{pdRDT2} = propagation delay of an RDT2 circuit element

t_{suRDT2} = setup time of an RDT2 circuit element

t_{pdEXOR} = propagation delay of an EXOR-gate

$t_{pdGFADD}$ = propagation delay of a GF(2^m) addition element

$t_{pdGFMULT}$ = propagation delay of a GF(2^m) multiplier element

As stated in chapter 3 the longest path through a serial LFSR generator polynomial encoder is two addition elements, a multiplier and a storage device. Hence the minimum clock period for a serial circuit is given by

$$t_{CLKMINs_{(Non-binary)}} = t_{suRDT2} + t_{pdRDT2} + 2t_{pdGFADD} + t_{pdGFMULT} \tag{4.17}$$

From Appendix A we find that $t_{pdGFMULT}$ is $3 \times t_{pdEXOR}$ and $t_{pdGFADD} = t_{pdEXOR}$ for a codeword using 8-bits/symbol. Hence we have

$$t_{CLKMINs_{(Non-binary)}} = t_{suRDT2} + t_{pdRDT2} + 5t_{pdEXOR} \tag{4.18}$$

Likewise for the binary code we find that

$$t_{CLKMINs_{(Binary)}} = t_{suRDT2} + t_{pdRDT2} + 2t_{pdEXOR} \tag{4.19}$$

Since each circuit must be clocked a total of k times in order to generate the $(n-k)$ check bits/symbols, the total computational time T_{Cs} is given by

$$T_{Cs_{(Non-binary)}} = k \times t_{CLKMINs_{(Non-binary)}} \tag{4.20}$$

$$T_{Cs_{(Binary)}} = k \times t_{CLKMINs_{(Binary)}} \tag{4.21}$$

Intermediate circuit solutions:

Figure 4.8 illustrates the general form of an intermediate circuit solution for check-symbol/syndrome generation. The serial input data is demultiplexed into p-symbol wide data words. They are then presented to a combinational logic circuit which is configured to perform all necessary addition and multiplication according to the relationships derived from the appropriate transition matrix. With each successive clock cycle the output from this circuit is stored in a $(n-k)$ bit/symbol wide parallel latch which provides feedback for the following cycle. After s clock cycles the final check-symbol/syndrome values are latched onto the output of the circuit. Having outlined the operational nature of the system it is now possible to derive expressions for both the computational time and complexity.

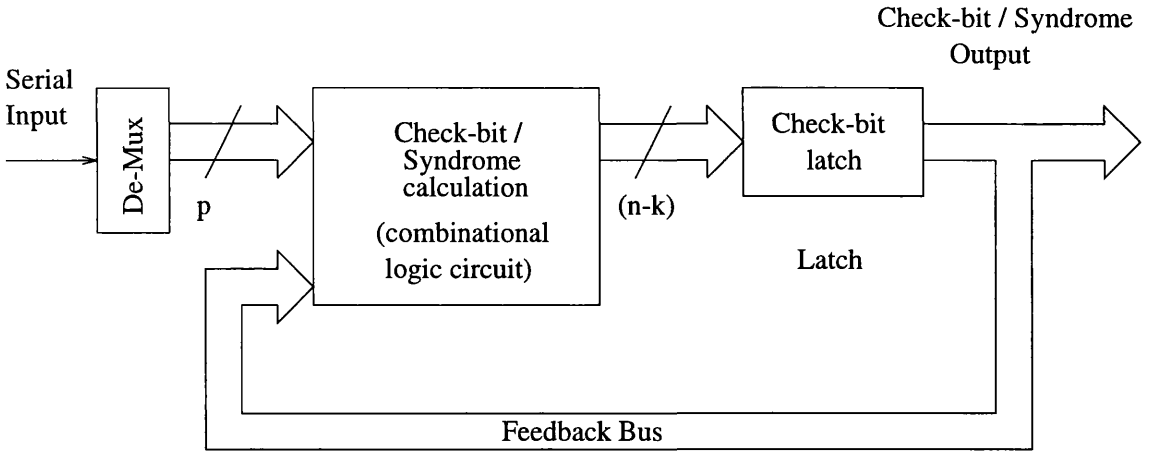


Figure 4.8: General form of intermediate circuit.

By inspection of the relevant transition matrix we find that the combinational logic circuit consists of N_{MUL} multipliers and N_{ADD} adders. The total system complexity (ignoring the demultiplexing) is given by

$$N_{Ti_{(Non-binary)}} = N_{MUL} \times \text{Multipliers} + m(n-k) \times \text{RDT2s} + N_{ADD} \times \text{Adders}$$

$$N_{Ti_{(Non-binary)}} = 224N_{MUL} + 48(n-k) + 32N_{ADD} \tag{4.22}$$

And for the binary case:

$$\begin{aligned} N_{Ti_{(Binary)}} &= N_X \times \text{EXOR} - \text{gates} + (n-k) \times \text{RDT2s} \\ N_{Ti_{(Binary)}} &= 4N_X + 6(n-k) \end{aligned} \tag{4.23}$$

where N_X is the number of EXOR-gates used in the check bit calculation circuit.

We proceed to calculate the minimum latching time of the intermediate circuits by noting that in general, for non-binary codes, the output from each of the $(n-k)$ latches is often multiplied by a constant in GF(2^m). These products are subsequently added to further products to form the outputs of the combinational logic circuit. Since all multiplications may be performed in parallel and assuming that

the demultiplexing time is negligible, the minimum latching time is found to be

$$\begin{aligned} t_{CLKMINi_{(Non-binary)}} &= t_{suRDT2} + t_{pdRDT2} + t_{pdGFMULT} \\ &\quad + N_{ADD_{(Stages)}}\, t_{pdEXOR} \end{aligned} \tag{4.24}$$

where $N_{ADD_{(Stages)}}$ is the number of addition stages. For binary codes the minimum clock period is given by

$$t_{CLKMINi_{(Binary)}} = t_{suRDT2} + t_{pdRDT2} + N_{X_{(Stages)}}\, t_{pdEXOR} \tag{4.25}$$

where $N_{X_{(Stages)}}$ is the number of EXOR-stages. In the case of the intermediate circuits the total computational time, T_{Ci}, is given by

$$T_{Ci_{(Non-binary)}} = s \times t_{CLKMIN_{i\,(Non-binary)}} \tag{4.26}$$

$$T_{Ci_{(Binary)}} = s \times t_{CLKMIN_{i\,(Binary)}} \tag{4.27}$$

From the previous equations it is now possible to calculate the complexity and check-symbol computation time for any intermediate circuit solution.

Parallel circuits:

The parallel implementation of the circuit is similar to that of the intermediate case, except that no output latching registers are required to store the check symbols for each clock cycle. In this case the serial data is demultiplexed into a k symbol wide parallel word which is fed into the combinational logic circuit. This circuit comprises of a number of Galois field adders and multipliers configured, according to the transition matrix, to produce the $(n-k)$ check symbols for each k symbol word. Proceeding as for the generalised intermediate case we find that the complexity of the circuit, N_{Tp} is given by

$$N_{Tp_{(Non-binary)}} = N_{MUL} \times \text{Multipliers} + N_{ADD} \times \text{Adders}$$

$$N_{Tp_{(Non-binary)}} = 224 \times N_{MUL} + 32 \times N_{ADD} \tag{4.28}$$

and

$$\begin{aligned} N_{Tp_{(Binary)}} &= N_X \times \text{EXOR} - \text{gates} \\ &= 4N_X \end{aligned} \tag{4.29}$$

with the minimum computation times given by

$$T_{Cp_{(non-binary)}} = t_{pdGFMULT} + N_{ADD_{(Stages)}}\, t_{pdEXOR} \tag{4.30}$$

$$T_{Cp_{(Binary)}} = N_{X_{(Stages)}}\, t_{pdEXOR} \tag{4.31}$$

4.2.4 Parity polynomial based circuits.

Section 3.2 outlined how the generation of codewords may be achieved by employing circuits based on a code's parity polynomial. These circuits operate in a similar manner to generator polynomial based encoders and as such are subject to series-parallel techniques.

In the case of a code with parity polynomial $\boldsymbol{h}(X) = h_0 + h_1X + ... + h_{k-1}X^{k-1} + h_kX^k$ the following $(k \times k)$ transition matrix may be defined which describes the circuit of figure 4.9

$$\mathrm{T} = \begin{bmatrix} 0 & I_{k-1} & & 0 \\ \vdots & & \ddots & \\ 0 & & & \\ h_0 & \dots & \dots & h_{k-1} \end{bmatrix} \tag{4.32}$$

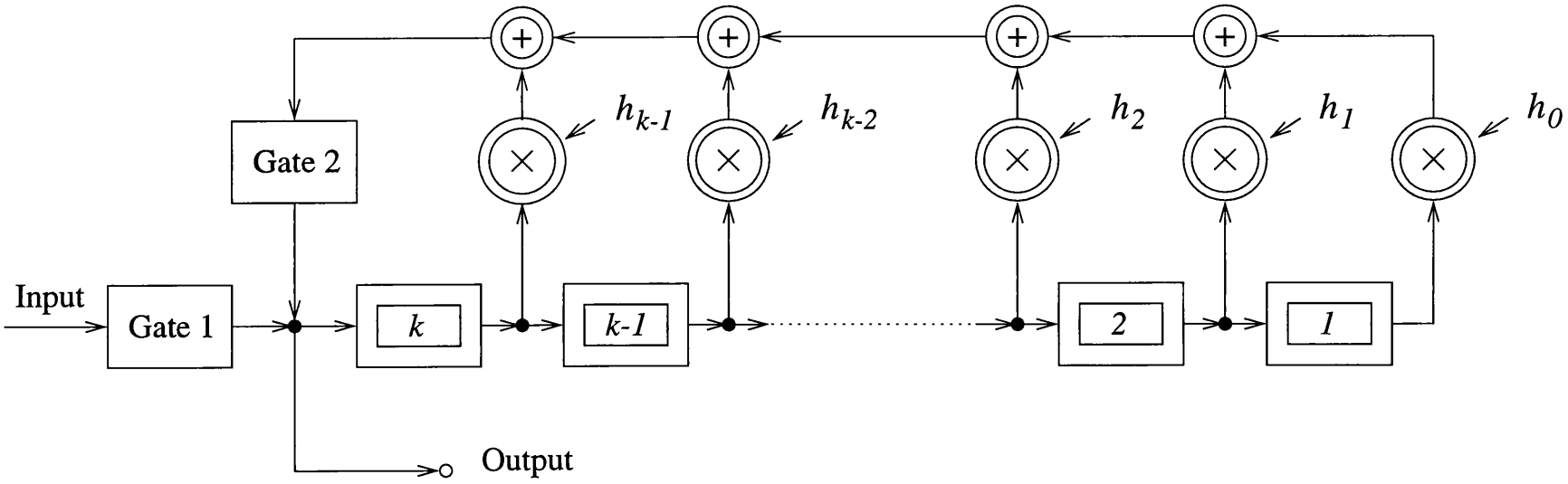


Figure 4.9: Autonomous view of a serial encoder based on the parity polynomial $\boldsymbol{h}(X)$.

Returning to the (7,4) Hamming code based on the parity polynomial $\boldsymbol{h}(X) = 1 + X + X^2 + X^4$ the following matrix may be constructed which describes the circuit of figure 4.10

$$\mathrm{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \tag{4.33}$$

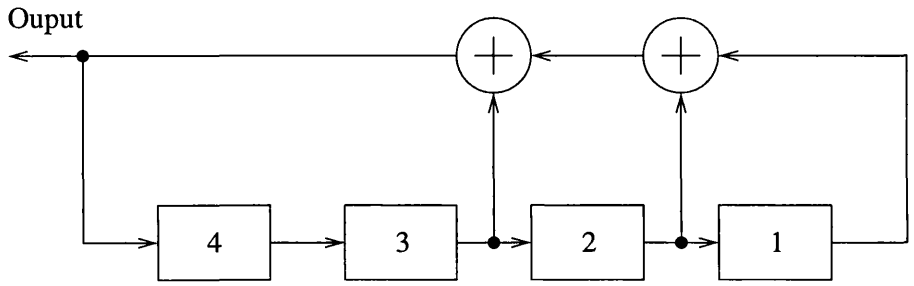


Figure 4.10: Serial Hamming (7,4) encoder based on $\boldsymbol{h}(X) = 1 + X + X^2 + X^4$.

Once loaded with the information vector, each successive clock cycle produces a single parity check digit. By raising T to the power 2 it is possible to produce a matrix which defines the circuit shown in figure 4.11. In this instance two parity check digits are produced per clock cycle.

$$\mathrm{T}^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \tag{4.34}$$

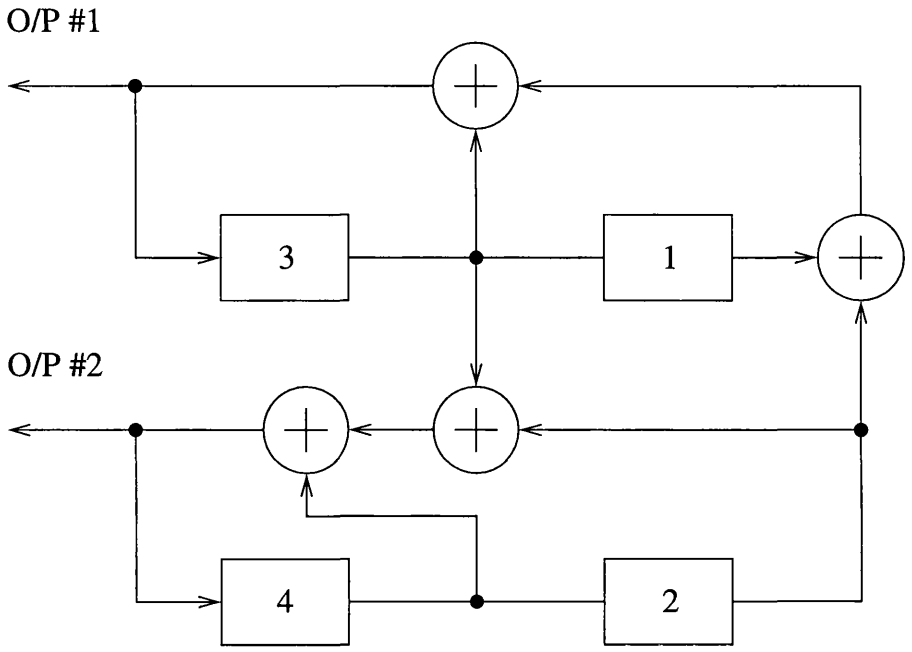


Figure 4.11: Series-parallel parity polynomial encoder based on T^2.

This circuit, although slightly more complex than that of figure 4.10, generates the parity check digits after only two clock cycles. We may note however that after two clock cycles we obtain four parity check digits, whereas we only require three; in this case the second digit produced by output #2 is ignored.

As with the generator polynomial example it is possible to further reduce the circuit to a single parallel stage by raising T to the power $(n-k)$ i.e. three. The result is the following matrix where the last three rows now represent the parity check equations.

$$T^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{4.35}$$

Therefore by raising T to the power $(n-k)$ we arrive at a circuit identical to that of

figure 4.7 which produces the parity check digits by direct extraction. Furthermore, as with the generator polynomial case, we have identified an intermediate circuit solution, T^2, which provides an alternative encoding structure. We continue once more by deriving expressions for the complexity and computational time for various circuit architectures based on the parity polynomial.

Serial parity polynomial circuits:

From figure 4.9 we observe that an encoder of this type requires k storage devices, $(n_h - 2)$ addition elements and $(n_h - 1)$ multipliers where n_h is the number of non-zero terms in the parity polynomial. The circuit complexity in terms of logic gates, N_{Ts}, is found to be

$$N_{Ts_{(Non-binary)}} = 48k + 224(n_h - 1) + 32(n_h - 2) \tag{4.36}$$

$$N_{Ts_{(Binary)}} = 6(n - k) + 4(n_h - 1) \tag{4.37}$$

with the minimum clock period given by

$$t_{CLKMINs_{(Non-binary)}} = t_{suRDT2} + t_{pdRDT2} + t_{pdGFADD}(n_h - 2) + t_{pdGFMULT}$$

$$t_{CLKMINs_{(Non-binary)}} = t_{suRDT2} + t_{pdRDT2} + t_{pdEXOR}(n_h + 1) \tag{4.38}$$

$$t_{CLKMINs_{(Binary)}} = t_{suRDT2} + t_{pdRDT2} + 2t_{pdEXOR} \tag{4.39}$$

Since circuits of this nature must be clocked $(n - k)$ times the total computational time, T_{Cs} is

$$T_{Cs_{(Non-binary)}} = (n - k) \times t_{CLKMINs_{(Non-binary)}} \tag{4.40}$$

$$T_{Cs_{(Binary)}} = (n - k) \times t_{CLKMINs_{(Binary)}} \tag{4.41}$$

Intermediate circuit solutions:

In the case of the intermediate circuit solutions we find that by inspection of the relevant transition matrix the circuit complexity is

$$\begin{aligned} N_{Ti_{(Non-binary)}} &= N_{MUL} \times \text{Multipliers} + mk \times \text{RDT2s} \\ &\quad + N_{ADD} \times \text{Adders} \\ N_{Ti_{(Non-binary)}} &= 224N_{MUL} + 48k + 32N_{ADD} \qquad (4.42) \\ N_{Ti_{(Binary)}} &= 4N_X + 6k \qquad (4.43) \end{aligned}$$

We also note that the expressions for the minimum latching times are identical to those of the generator polynomial circuits

$$\begin{aligned} t_{CLKMINi_{(Non-binary)}} &= t_{suRDT2} + t_{pdRDT2} + t_{pdGFMULT} \\ &\quad + N_{ADD_{(Stages)}}\, t_{pdEXOR} \qquad (4.44) \\ t_{CLKMINi_{(Binary)}} &= t_{suRDT2} + t_{pdRDT2} + N_{X_{(Stages)}}\, t_{pdEXOR} \qquad (4.45) \end{aligned}$$

However, unlike the intermediate circuit solutions based on the generator polynomial, in this instance each circuit must be clocked

$$s = \left\lceil \frac{(n-k)}{p} \right\rceil$$

times in order to generate the parity check digits, where $\lceil x \rceil$ denotes the smallest integer greater than x. The resulting computational time, T_{Ci}, is given by

$$\begin{aligned} T_{Ci_{(Non-binary)}} &= s \times t_{CLKMIN_{i\ (Non-binary)}} \qquad (4.46) \\ T_{Ci_{(Binary)}} &= s \times t_{CLKMIN_{i\ (Non-binary)}} \qquad (4.47) \end{aligned}$$

Parallel circuits:

As with the parallel generator polynomial based circuits , the parallel parity polynomial circuit does not require any storage elements to provide feedback for subsequent

clock cycles. Since the circuit consists only of a combinational logic circuit the complexity may be expressed as follows

$$\begin{aligned} N_{Tp_{(Non-binary)}} &= N_{MUL} \times \text{Multipliers} + N_{ADD} \times \text{Adders} \\ N_{Tp_{(Non-binary)}} &= 224N_{MUL} + 32N_{ADD} \end{aligned} \tag{4.48}$$

$$\begin{aligned} N_{Tp_{(Binary)}} &= N_X \times \text{EXOR} - \text{gates} \\ N_{Tp_{(Binary)}} &= 4N_X \end{aligned} \tag{4.49}$$

Likewise the computational time is

$$t_{CLKMINp_{(Non-binary)}} = t_{pdGFMULT} + N_{ADD_{(Stages)}}\, t_{pdEXOR} \tag{4.50}$$

$$t_{CLKMINp_{(Binary)}} = N_{X_{(Stages)}}\, t_{pdEXOR} \tag{4.51}$$

4.2.5 Series-parallel error detection

Having presented numerous encoding architectures based on both generator and parity polynomials this section now concludes by briefly introducing series-parallel syndrome calculation or error detection circuits. In this instance we shall only investigate solutions relating to binary codes as error detection circuits for non-binary codes, as previously stated, prove to be trivial cases.

As series-parallel techniques have already been demonstrated in some depth it is considered only necessary to illustrate the standard syndrome architecture and define the relevant transition matrix. Figure 4.12 shows the general form of a syndrome calculation circuit based on the minimal polynomial $\Phi(X) = 1 + \phi_1 X + \phi_2 X^2 + \ldots + \phi_m X^m$. It is clear that this circuit is almost identical to that of the polynomial division circuits based on the generator polynomial as shown in figure 4.4. The most noticeable difference being the received vector is fed into the circuit from the left hand side and that the syndrome components are now stored in only m registers.

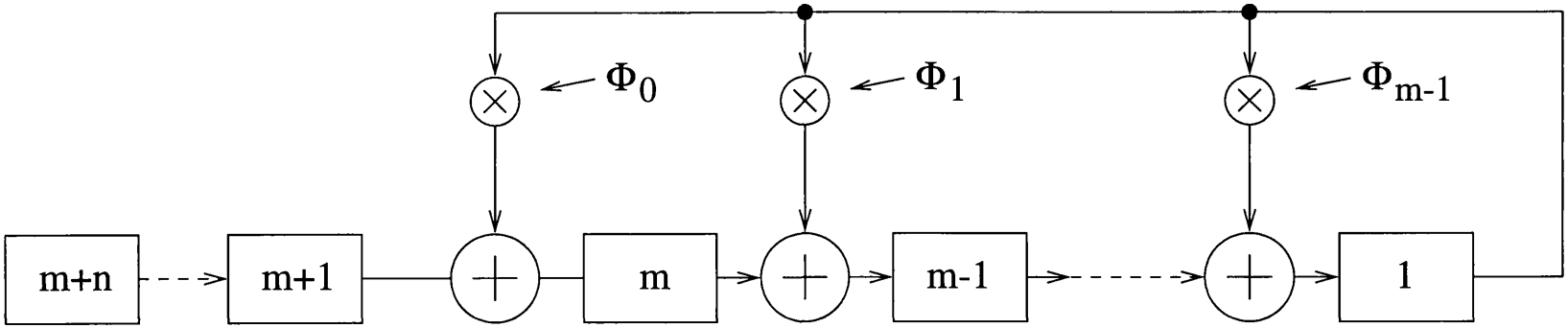


Figure 4.12: Autonomous view of a standard syndrome calculation circuit.

The functional specification of the previous circuit may be defined by the following $(n+m) \times (n+m)$ transition matrix.

$$\mathrm{T} = \begin{bmatrix} \phi_{m-1} & I_{m+n-1} & & \\ \phi_{m-2} & & \ddots & \\ \vdots & & & \\ \vdots & & & \\ \phi_0 & & & \\ 0_{(n)\times(m+n)} & \cdots & & \\ \vdots & & & \\ & & & 0 \end{bmatrix} \tag{4.52}$$

Having already developed general expressions for similar encoding circuits based on polynomial division the a summary of circuit complexities and computational times for various circuit types is given.

Serial polynomial division circuits:

Circuit complexity

$$\begin{aligned} N_{Ts_{(Binary)}} &= m \times \mathrm{RDT2s} + (n_\Phi - 1) \times \mathrm{EXOR} - \mathrm{gates} \\ &= 6m + 4(n_\Phi - 1) \end{aligned} \tag{4.53}$$

where n_Φ is the number of non-zero terms in the minimal polynomial.

Computational time

$$t_{CLKMINs_{(Binary)}} = t_{suRDT2} + t_{pdRDT2} + t_{pdEXOR} \tag{4.54}$$

Since each circuit must be clocked a total of n times in order to generate the m check bits, the total computational time T_{Cs} is given by

$$T_{Cs_{(Binary)}} = n \times t_{CLKMINs} \tag{4.55}$$

Intermediate circuit solutions:

Circuit complexity

$$\begin{aligned} N_{Ti_{(Binary)}} &= N_X \times \text{EXOR} - \text{gates} + m \times \text{RDT2s} \\ &= 4N_X + 6m \end{aligned} \tag{4.56}$$

Minimum clock period

$$t_{CLKMINi_{(Binary)}} = t_{suRDT2} + t_{pdRDT2} + N_{X_{(Stages)}}\, t_{pdEXOR} \tag{4.57}$$

At this point it is important to note that syndrome calculation circuits must be clocked exactly

$$s = \frac{n}{p}$$

times in order to generate the correct syndrome vector. The resulting computational time is given by

$$T_{Ci_{(Binary)}} = s \times t_{CLKMINi} \tag{4.58}$$

Parallel circuits:

Circuit complexity

$$\begin{aligned} N_{Tp_{(Binary)}} &= N_X \times \text{EXOR} - \text{gates} \\ &= 4N_X \end{aligned} \tag{4.59}$$

Computational time

$$T_{Cp_{(Binary)}} = N_{X_{(Stages)}}\, t_{pdEXOR} \tag{4.60}$$

4.3 Results

Using the equations derived in previous sections we now evaluate and summarise graphically each of the design solutions relating to the various codes considered. In this instance the codes investigated represent low complexity, high rate binary BCH codes which are of particular relevance to high bit rate optical transmission systems. Figures 4.13 to 4.18 relate to binary encoding architectures based on the generator and parity polynomials of the codes selected. Figures 4.14 and 4.16 are given to illustrate the performance and requirements for circuit architectures with practical applications i.e. degree of parallelism $\leq$ 64. In order to provide a comparison, figures 4.19 and 4.20 illustrate the results obtained from a similar study performed on the multi-level RS(255,239) code which is currently the focus of much interest in the field of fibre optic transmission systems. To conclude, the factors relating to syndrome calculation circuits for the corresponding binary codes are investigated and presented in figures 4.21 and 4.22. The generator, parity and minimal polynomials used in this exercise can be either found in, or derived from [20].

In each case the computational time and equivalent circuit complexity is plotted for varying degrees of parallelism thereby highlighting the effective trade-off between circuit speed and complexity. It may be noted that in each case the two extremities of each plot represent the values obtained for the classical serial and parallel implementation of the encoding/syndrome calculation circuits . Intermediate points represent the extent to which a compromise may be effected by appropriate selection of an intermediate-type circuit.

To allow comparisons to be drawn between the various logic families the computational time for each of the figures shown has been normalised to the propagation delay, t_{pdNAND}, of a single three input NAND-gate.

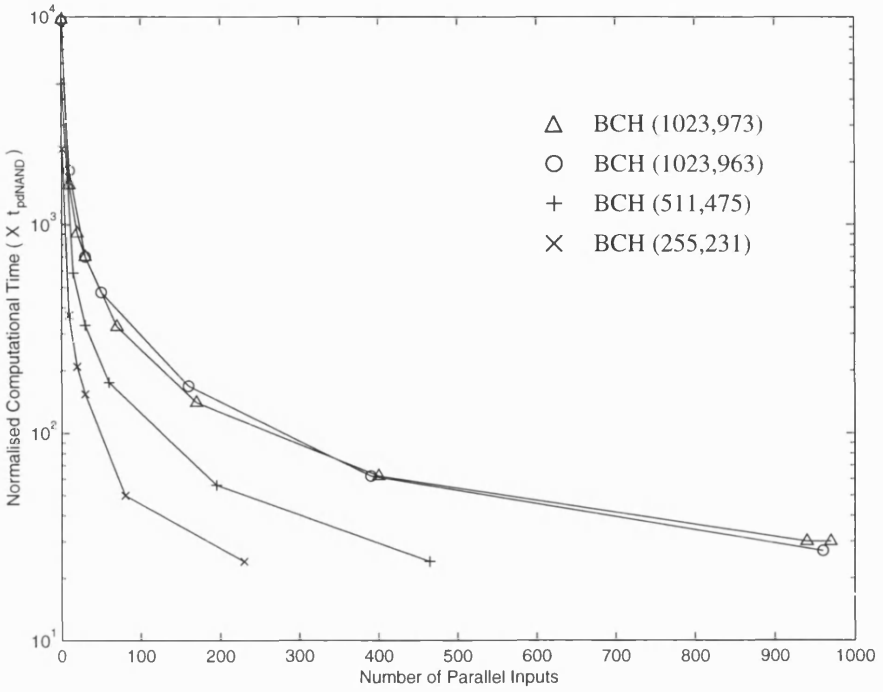


Figure 4.13: Computational time vs. degree of parallelism - generator polynomial based encoders.

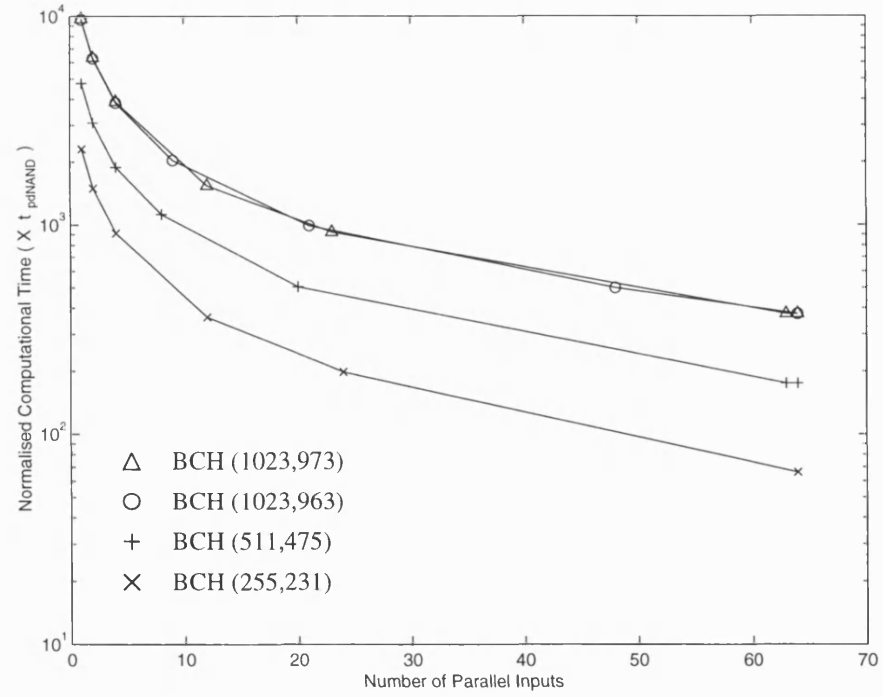


Figure 4.14: Computational time vs. degree of parallelism ≤ 64 - generator polynomial based encoders.

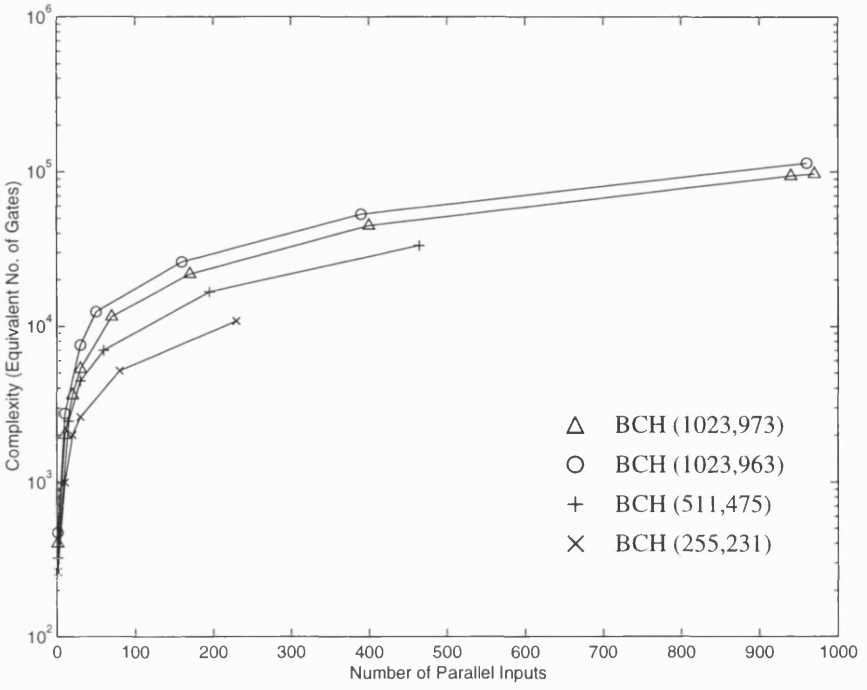


Figure 4.15: Complexity vs. degree of parallelism - generator polynomial based encoders.

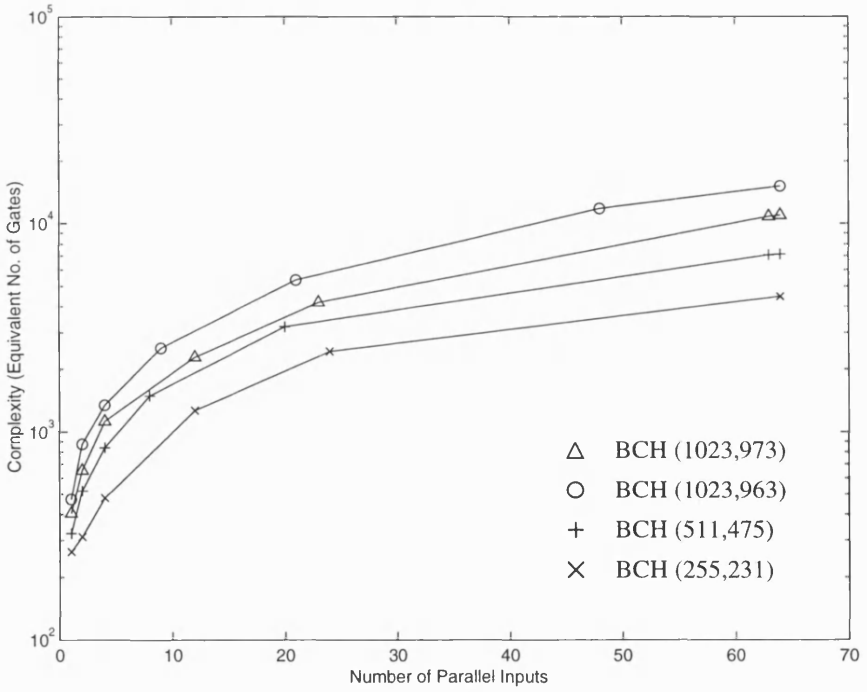


Figure 4.16: Complexity vs. degree of parallelism ≤ 64 - generator polynomial based encoders.

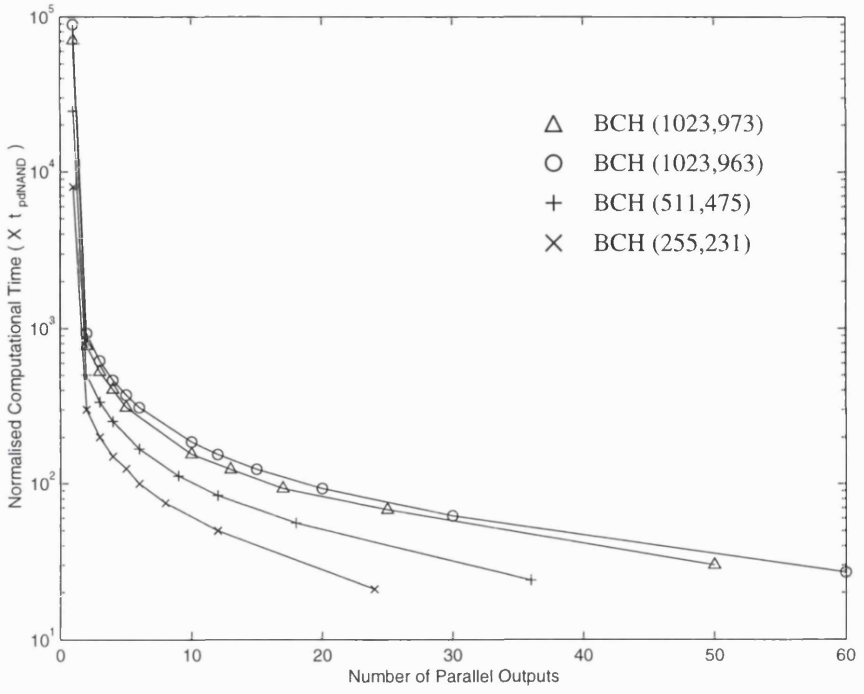


Figure 4.17: Computational time vs. degree of parallelism - parity polynomial based encoders.

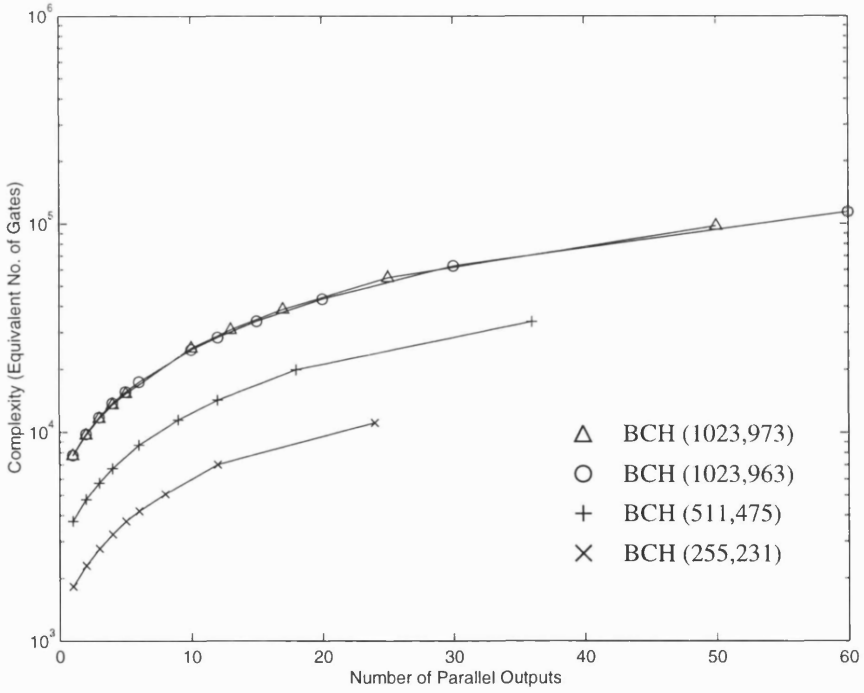


Figure 4.18: Complexity vs. degree of parallelism - parity polynomial based encoders.

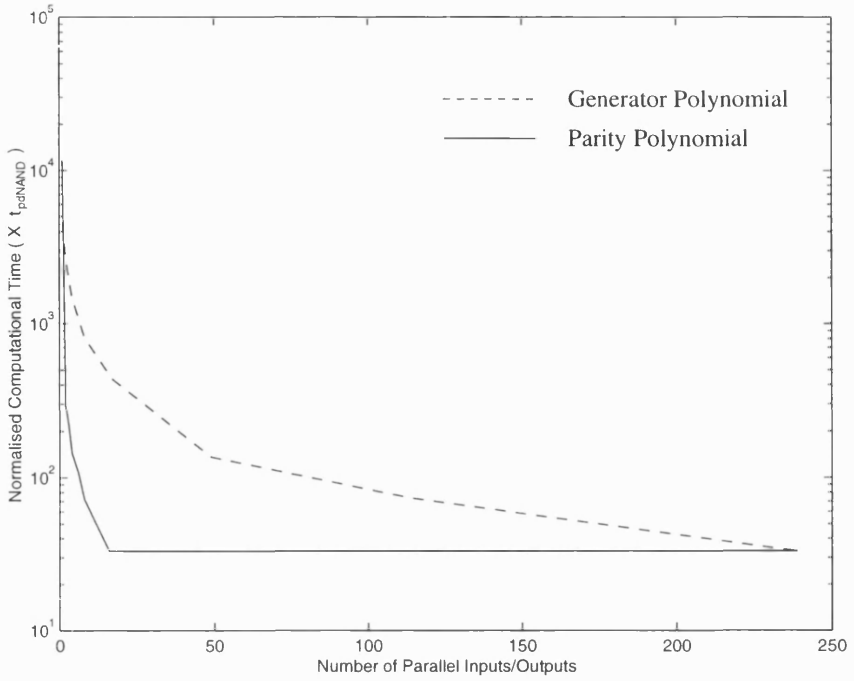


Figure 4.19: Comparative analysis of computational time between generator and parity polynomial based encoders for the RS(255,239) code.

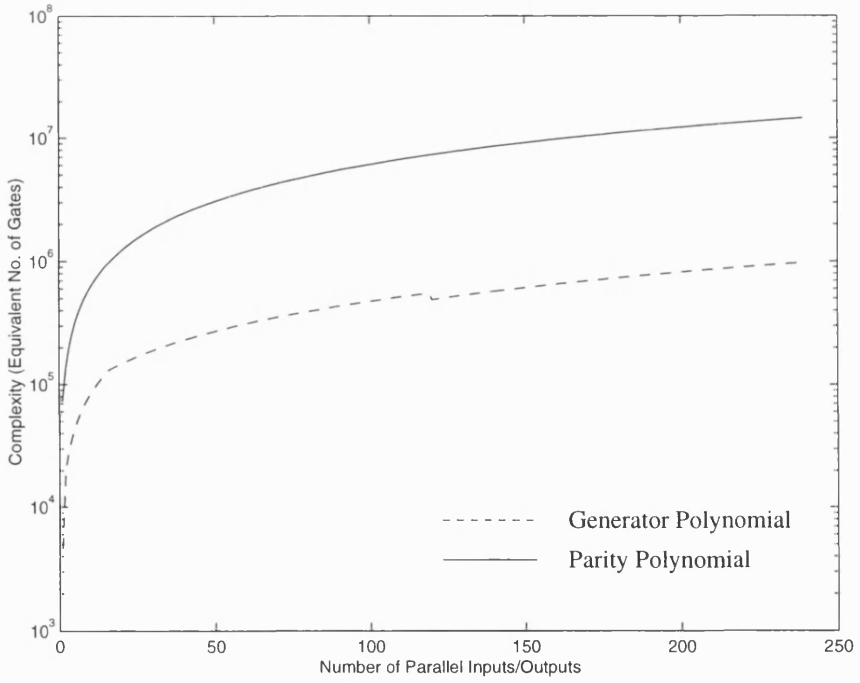


Figure 4.20: Comparative analysis of circuit complexity between generator and parity polynomial based encoders for the RS(255,239) code.

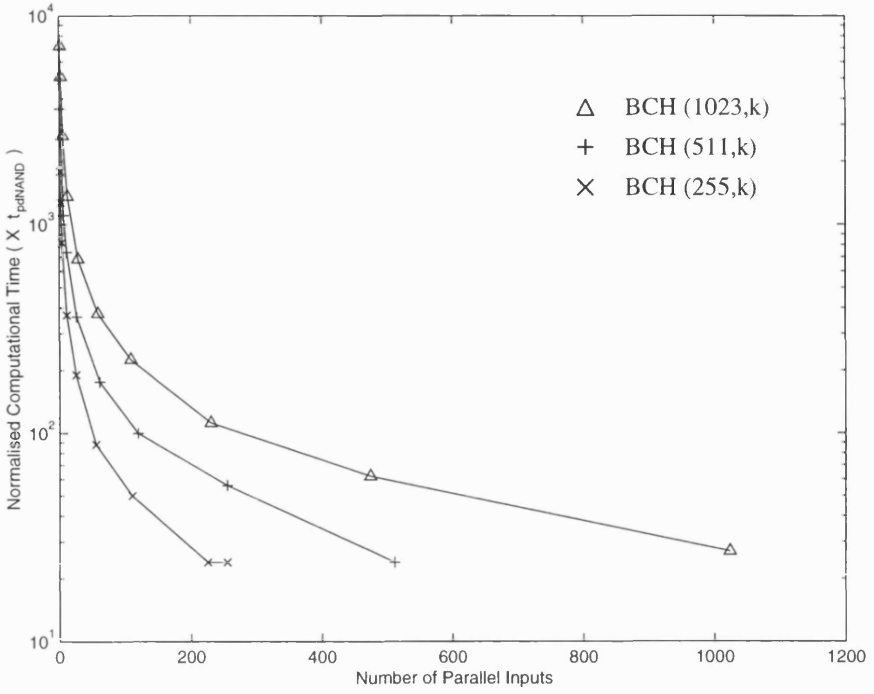


Figure 4.21: Computational time vs. degree of parallelism - syndrome calculation circuits.

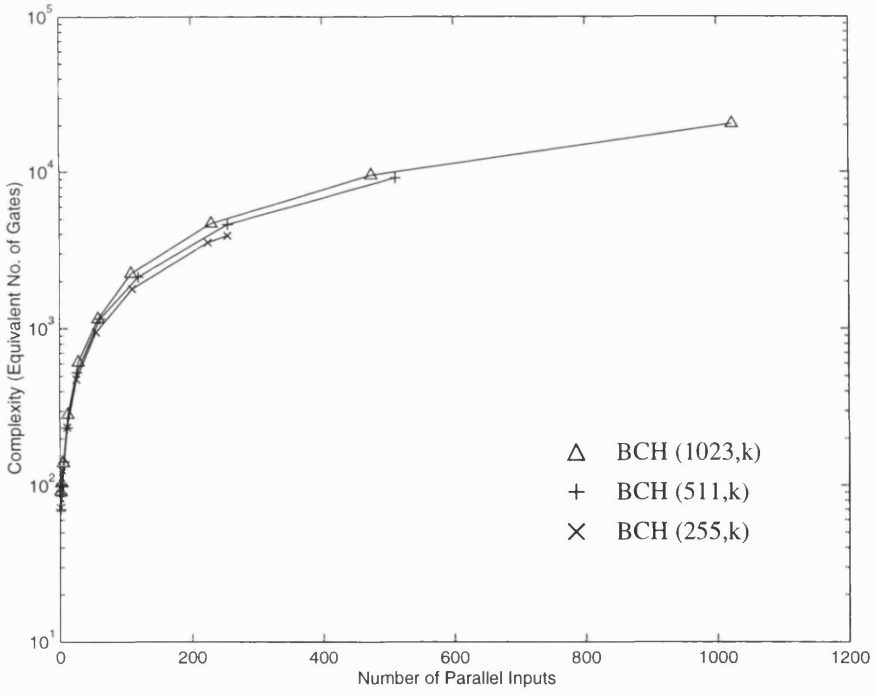


Figure 4.22: Complexity vs. degree of parallelism - syndrome calculation circuits.

It may be observed that in all cases the computational times range from 10^4 for a serial implementation to approximately 30 for fully parallel arrangement. For a typical value of $t_{pdNAND} = 10ns$, which assumes a TTL type fabrication process, the resulting figures correspond to average bit rates ranging from 1 Mbit/s to approximately 3.5 Gbit/s for binary codes and 20 MBit/s to 6.8 GBit/s for the RS code. Although appearing to offer superior performance in terms of speed over the binary encoders, the RS encoder requires almost a ten fold increase in the number of equivalent gates required for implementation. In the case of error detection figure 4.21 indicates that computational times for syndrome calculation may be matched to anticipated encoding speeds.

Further increases in bit speed may be obtained by the selection of a different process in which to fabricate the encoder/syndrome calculation circuits. At this point it is important to stress that the construction of certain circuit elements such as D-type flip-flops vary between individual logic families such as ECL and TTL [35] [36]. These differences must therefore be considered when calculating subsequent computational time and complexity curves although the functional specification of the circuit remains constant.

Turning to the binary encoding arrangements the results indicate comparable performance between parity and generator polynomial based circuits. Although as expected the parity polynomial approach requires a greater number of storage elements. This fact is reflected in figures 4.16 and 4.18 where, assuming the required degree of parallelism is 8, the parity polynomial circuit is approximately 10 times more complex than that of the generator polynomial circuits.

This would initially suggest that polynomial division or generator polynomial based circuits are best suited to series-parallel implementation. However this method could present problems when multiplexing to the required line rate. In this instance all

$(n-k)$ parity check digits become simultaneously available after the final clock cycle, which for the longer codes represents a large number of bits. In contrast, if a modest increase in complexity can be tolerated then parity polynomial based circuits can be employed with intermediate circuit configurations producing p check bits per clock cycle. With this approach the resulting circuit is found to be much more amenable to the multiplexing process since only p bits per clock cycle require multiplexing. Alternatively this circuit configuration could directly drive optical time division multiplexing circuitry thereby expunging the need for electrical multiplexers [37] [38].

4.4 Summary

In this chapter we have examined new series-parallel arrangements for encoding and error detection. Following standard architectures which rely on established serial of fully parallel circuits new binary and non-binary series-parallel encoding architectures were presented and their performance quantified. This was followed by an exercise of the appliance of series-parallel techniques to error detection where again beneficial trade offs relating to speed and circuit complexity were noted and quantified. In addition to this a design process has been presented which enables various circuit arrangements to be generated with varying degrees of parallelism. Since the different circuit configurations are generated from the initial functional specification of a serial circuit this process is directly adaptable to all logic families.

This chapter provides the initial basis for the prospective delivery of very high speed FEC systems with practical applications to high bit rate optical communications. These applications aspects will be addressed separately in chapter 6. However, it should be noted that whilst we have presented circuits for encoding and error detec-

tion a full FEC system calls for high speed error correction, which has not proved directly adaptable to series-parallel realisation. To address this we will examine, in the next chapter, how the series-parallel error detection schemes devised and discussed here may be employed in a buffered decoding arrangement to achieve very high bit rate operation.

Chapter 5

Buffered Decoding

5.1 Introduction

The concept of series-parallel circuit architectures has been identified in the previous chapter as a technique for obtaining high speed encoding and error detection in a digital communications system. However for systems which require error correction in addition to error detection this technique does not readily extend to the error correction circuitry since the series-parallel structures involved pertain only to circuits consisting of feedback shift register operations. This chapter illustrates how the series-parallel encoding and error detection strategy can be utilised with the concept of buffered decoding [39] to realise high speed forward error correction (FEC).

In this chapter we examine two variations of the buffered decoding arrangement. In the first instance we consider a decoder which decodes high rate binary codes that are of particular relevance to high speed digital systems. By performing a numerical analysis it is possible to calculate the increase in speed afforded by such a system and

the buffer lengths required in order to prevent buffer overflow. We then investigate a buffered decoder which incorporates a standard commercial RS decoding IC as the main error correction module. By performing a similar analysis to that of the binary decoder we demonstrate the potential, in terms of an overall increase in decoding speed, offered by this approach. Initially we begin by introducing the general concepts of buffered decoding.

5.2 Buffered Decoding

When systematic codewords are used in an FEC system the decoding process can be divided into two separate stages; error detection (syndrome calculation) followed by error correction. The error detection stage is by far the simplest of these two operations since it only requires the use of LFSR circuits and, as demonstrated in chapter 3, can be configured to operate in a series-parallel form. In contrast, the error correction stage is far more complex and utilises the output from the error detection circuitry (the syndromes) to determine the error pattern and subsequently correct the received word. However in most systems the majority of received words are error free and therefore do not need error correction; the second stage of decoding can thus be by-passed for these cases. It is this property which the buffered decoding concept uses to speed-up the decoding process by enabling the decoder to work at an average speed rather than a worst case speed. This can be accomplished by buffering the received words following error detection; provided the buffers are long enough then this does not contribute a significant amount to an increase in output bit error rate when buffer overflow occurs.

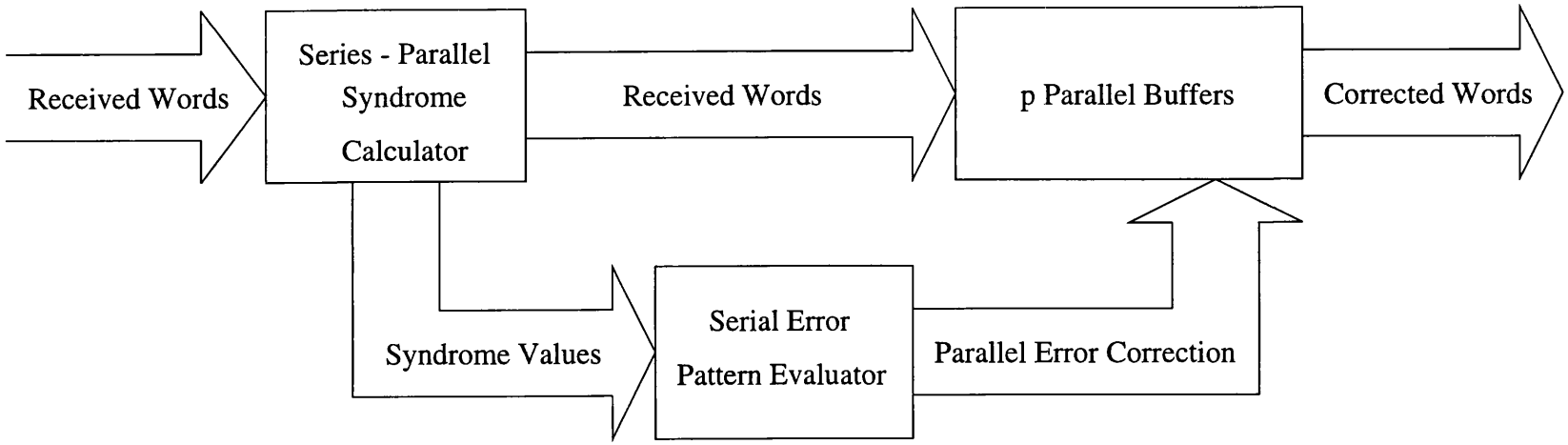


Figure 5.1: Buffered decoding system incorporating series-parallel error detection.

Figure 5.1 shows a block diagram of the proposed system where syndrome calculation/error detection is carried out in series-parallel form with only corrupt words being processed by the serial error pattern evaluator.

The key parameters in determining the performance of a buffered decoding system are: B - the buffer length, T - the inter-arrival time between codewords (the number of machine cycles required for a complete codeword to enter the buffer, which is assumed to be equal to the number of machine cycles required to calculate the syndrome $= n$, the codeword length in a serial decoding system), λ - the average number of channel errors per codeword and M_i the number of machine cycles required to decode a codeword containing i errors.

Following [39] we model the number of errors in a codeword as a Poisson random variable with mean λ, thus the average decoding time (in machine cycles) is:

$$\bar{M}(\lambda) = \sum_{i=0} M_i \mathrm{e}^{-\lambda} \frac{\lambda^i}{i!} \tag{5.1}$$

If we denote D_n as the decoding time required by the n-th codeword and $d(j)$ as the probability that the n-th codeword takes j cycles to decode then we may assume:

$$\bar{D}_n = \sum_{j} j d(j) < T \tag{5.2}$$

i.e. the average decoding time is less than the time required for one codeword to enter the buffer. This simply ensures that on average the decoder can keep ahead of the traffic.

Using the standard notation of queueing theory, a system such as the one shown in figure 5.1 is referred to as a D/G/1 queue. Where "D" means that the arrival time of codewords is deterministic, i.e. constant; "G" means that the codeword service times are general, i.e arbitrary; and "1" means that there is only one server

or decoder. By analysing this queue in a similar manner to a G/G/1 queue [40] it has been demonstrated [39] that:

$$\text{Prob}\{BO(n)\} \leq r_0^{B-T+1} \tag{5.3}$$

where $\text{Prob}\{BO(n)\}$ denotes the probability that the n-th codeword causes buffer overflow and r_0 is the smallest positive real root greater than unity of the following equation:

$$D^*(Z) - Z^T = \sum_{j=0} d(j)Z^j - Z^T = 0 \tag{5.4}$$

where $D^*(Z)$ is the probability generating function of $d(j)$.

From equation (5.3) we can deduce that $\text{Prob}\{BO(n)\} \leq 10^{-\gamma}$ if

$$B \geq T + \frac{\gamma}{\log_{10} r_0} - 1 \tag{5.5}$$

5.3 Approximations of Buffer Length

The analysis in the previous section provides a means for estimating the required buffer length for a buffered decoding system such that the buffering process does not significantly contribute to an increase in output bit error rate. Given a particular coding scheme the main factors affecting the buffer lengths are; λ, M_i and γ - the additional errors caused by buffer overflow. We shall consider two cases; (1) a worst case condition where it is assumed that all words in error require the same number of machine cycles to decode and (2) an optimised decoder which treats single error patterns as special cases and allows them to be decoded much faster than when 2 or more errors occur.

5.3.1 Worst case decoder

Here we assume that when errors occur $M_i = M$ for all $i \geq 1$. Since the decoder may be bypassed entirely if a zero syndrome is detected we may also assume $M_0 = 0$. This enables considerable simplification to be made to the equations in the previous sections and may be usefully employed to establish an upper bound on buffer length given a maximum number of machine cycles to decode. With this assumption the equations presented in the previous section simplify as follows:

$$\begin{aligned} \bar{M}(\lambda) &= M(1 - \mathrm{e}^{-\lambda}) \\ &\approx M\lambda \quad \text{for small } \lambda \end{aligned} \tag{5.6}$$

This implies that the increase in speed provided by the buffered decoding system is $\propto 1/\lambda = 1/nP_{ce}$, where n is the codeword length and P_{ce} is the channel bit error rate.

From equation(5.2) we find:

$$M(1 - \mathrm{e}^{-\lambda}) < T \tag{5.7}$$

i.e. M is upper bound by

$$M < \frac{T}{1 - \mathrm{e}^{-\lambda}} \tag{5.8}$$

Similarly equation (5.4) simplifies to:

$$D^*(Z) - Z^T = \mathrm{e}^{-\lambda} + (1 - \mathrm{e}^{-\lambda})Z^M - Z^T = 0 \tag{5.9}$$

By employing these simplified equations an indication of the length of buffers required for particular coding schemes may be obtained using M as a variable parameter.

5.3.2 Optimised decoder

As previously mentioned, certain algorithms exist which allow single errors to be decoded much faster than when multiple errors occur. By exploiting this feature a buffered decoder can be designed which operates at an increased average decoding speed.

Optimised decoders use these algorithms as a means by which the detection of certain error patterns allows the decoder to enter an alternative error correction procedure to that which is normally used. In doing so we can effectively decrease the average decoding time and therefore reduce the lengths of the required buffers. To illustrate this we have assumed that single errors may be corrected faster than when 2 or more errors occur. We shall now introduce a new variable called the optimisation parameter, β, where β is the ratio of the number of machine cycles required to correct 1 error to the number of machine cycles required to correct 2 or more errors. Figure 5.2 illustrates an optimised buffered decoder where both single and multiple error correction is performed in parallel.

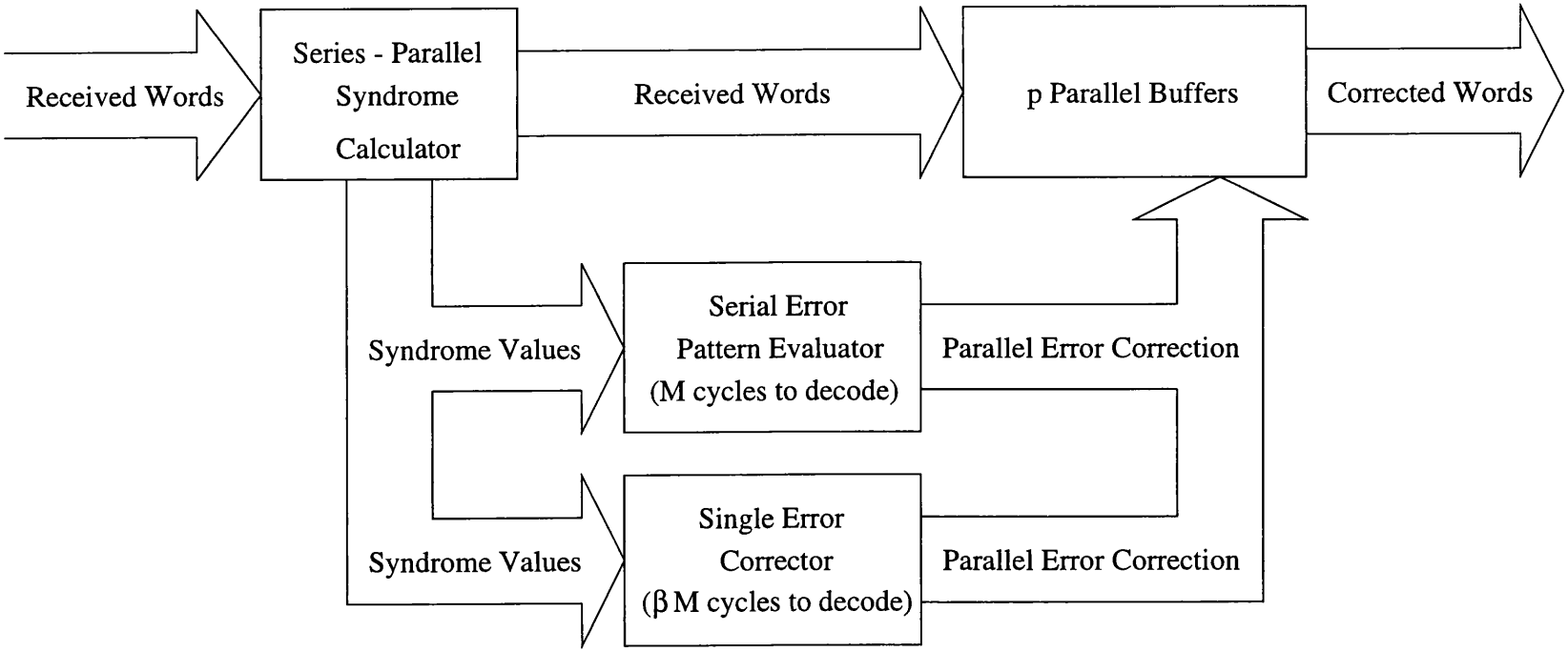


Figure 5.2: Optimised buffered decoding system.

If M_i denotes the number of machine cycles required to decode i errors then we obtain the following.

$$\begin{aligned} M_0 &= 0 \\ M_1 &= \beta \mathrm{M} \\ M_{>1} &= \mathrm{M} \end{aligned}$$

where $0 < \beta < 1$ for an optimised decoder. Substituting the above values into equation (5.2) and (5.4) yields the following:

$$\beta M \mathrm{e}^{-\lambda}\lambda + M(1 - \mathrm{e}^{-\lambda} - \mathrm{e}^{-\lambda}\lambda) < T \tag{5.10}$$

and

$$D^*(Z) - Z^T = \mathrm{e}^{-\lambda} + (\lambda \mathrm{e}^{-\lambda})Z^{\beta M} + (1 - \mathrm{e}^{-\lambda} - \lambda \mathrm{e}^{-\lambda})Z^M - Z^T = 0 \tag{5.11}$$

Re-arranging equation (5.10) we again place and upper bound on M thus

$$M < \frac{T}{1 - \mathrm{e}^{-\lambda} + \mathrm{e}^{-\lambda}\lambda(\beta - 1)} \tag{5.12}$$

It can now be seen that for any given value of optimisation i.e. β, we have an upper bound placed on the value of M/T such that

$$\frac{M}{T} < \frac{1}{1 - \mathrm{e}^{-\lambda} + \mathrm{e}^{-\lambda}\lambda(\beta - 1)} \tag{5.13}$$

This suggests that for any given value of β error correction must be achieved within a certain number of machine cycles in order for the decoder to keep up on average.

We are now in a position to calculate values for r_0 and hence from equation (5.5) the buffer lengths required for various values of β, M and T. One of the most efficient and accurate methods for calculating the roots of equation (5.11) is the Newton-Raphson method [41] (See Appendix B). As an example we shall consider a (1023,943) $t = 8$ code and assume a decoded bit error probability of 10^{-10} (i.e.

$\lambda = 0.536$ and $\gamma = 11$ in order to make the contribution from buffer overflow negligible). Using the ratio M/T as a variable figure 5.3 illustrates the buffer lengths required for several values of β.

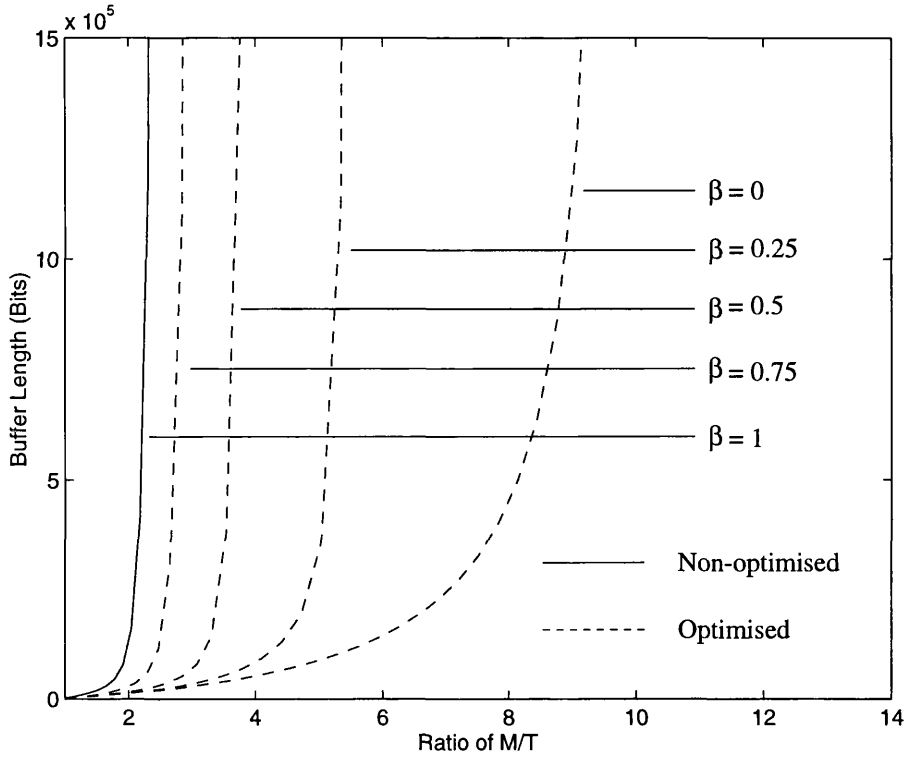


Figure 5.3: Buffer lengths as functions of M/T and β.

We may note from figure 5.3 that an upper limit for the ratio of M/T exists which is defined by the $\beta = 0$ curve. It is also important to note that for each curve a limiting value of M/T is given by

$$\frac{1}{1 - e^{-\lambda} + e^{-\lambda}\lambda(\beta - 1)} \tag{5.14}$$

which is a result of equation (5.13).

5.3.3 Single Error Optimisation

Optimisation for single error correction may be achieved in a number of ways. When used to correct single errors two of the most efficient, in terms of the number of machine cycles required to decode, are the Meggitt decoder [42] and Error Trapping decoder [43] [44]. Although very similar to implement there are subtle differences between the two techniques. The principles of both however rely on the cyclic nature of BCH codes and the relationship between the syndromes and correctable error patterns.

It has been shown [20] that there exists a relationship between the the syndrome $s(X)$ corresponding to the error pattern $e(X)$ and the i-th cyclic shift of the syndrome $s(X)^i$ corresponding to the ith cyclic shift of the error pattern $e(X)^i$. Therefore by storing the syndromes which correspond to all error patterns that contain an error the highest order bit position it is possible to cyclically shift the received syndrome until a match is found. Although theoretically simple the decoder is required to store

$$1+\sum_{i=1}^{t-1}\binom{n-1}{i}=1+\binom{n-1}{1}+\binom{n-1}{2}+\cdots+\binom{n-1}{t-1} \tag{5.15}$$

syndromes for a binary (n,k) cyclic code capable of correcting t errors. Although this value may be reduced by various algebraic techniques it is still prohibitively large for multiple error correcting codes with large block lengths. By restricting the decoder to search only for a syndrome which corresponds to a single error contained in the highest order bit position, i.e. single error correction, we need only store one syndrome value. This may be hard wired into the decoder thereby dramatically reducing the storage requirements. The time taken for the error to be located and corrected depends on the position of the error but at most will only require n shifts of the syndrome register.

Again by limiting the types of error patterns we intend to correct a Meggitt decoder may be practically implemented in the form of an Error Trapping decoder. If the errors are confined to $(n-k)$ consecutive bit positions (including the end-around case) then the errors may be trapped in the syndrome register if the weight of the syndrome is found to be t or less [27]. This guarantees the correction of all single errors and a good percentage of multiple errors. In this form the decoder need only determine the weight of the syndrome and as such is not required to store any syndrome values. The decoding time is once again dependent on the error location but like the Meggitt decoder requires at most n machine cycles.

5.4 Implementation of Decoder

In the initial analysis of the buffered decoding system it was assumed that both error detection and correction circuitry could be driven by a common system clock extracted from the incoming data e.g. 2.5 GHz for high speed systems. However, without resorting to the more esoteric logic families such as GaAs, the demand for such high speeds is beyond the current limits of digital logic.

One solution is to construct error detection and correction circuitry operating at slower speeds. Chapter 3 has already demonstrated that error detection may be realised in the multi gigabit region by use of series-parallel circuit techniques. By de-multiplexing down by a factor of eight for example, it has been shown that a S/P error detector operating at 320 Mb/s may achieve an overall data rate of 2.5 Gb/s. Since error detection may be performed in real time the inter-arrival time between codewords remains unaltered and therefore does not effect the operation of the buffered decoder.

Unfortunately error correction must be performed in a serial manner and is consequently limited in operational speed. Current logic families such as ECL can easily support speeds of 320 Mb/s, furthermore it has been demonstrated that decoders capable of operating at speeds comparable to this figure are realisable [45] [46].

Figure 5.4 illustrates a buffered decoder with error detection and correction being performed at a lower rate to that of the incoming data, timing buffers are used to synchronise the syndromes and error patterns with the decoder and delay line respectively.

In the previous example we have assumed that the buffered decoder operates on a 2.5 Gb/s data stream. De-multiplexing by a factor of eight results in each of the eight tributaries entering the delay line at 320 Mb/s (this does not appear to be a problem [47]). Assuming that current technology will allow conventional serial decoding at the tributary rate then for a decoder which previously required M cycles to decode this figure is now adjusted to $8 \times M$ cycles. Providing the limitation imposed by equation (5.2) is not violated then a buffered decoder may be successfully implemented. In general if we de-multiplex into p parallel tributaries then the error correction figure must be corrected to pM machine cycles. The resulting equations defining the behaviour of the decoding system are now given by

$$\bar{M}(\lambda) = \beta pM e^{-\lambda} + pM(1 - e^{-\lambda} - \lambda e^{-\lambda}) \tag{5.16}$$

and

$$D^*(Z) - Z^T = e^{-\lambda} + (\lambda e^{-\lambda}) Z^{\beta pM} + (1 - e^{-\lambda} - \lambda e^{-\lambda}) Z^{pM} - Z^T = 0 \tag{5.17}$$

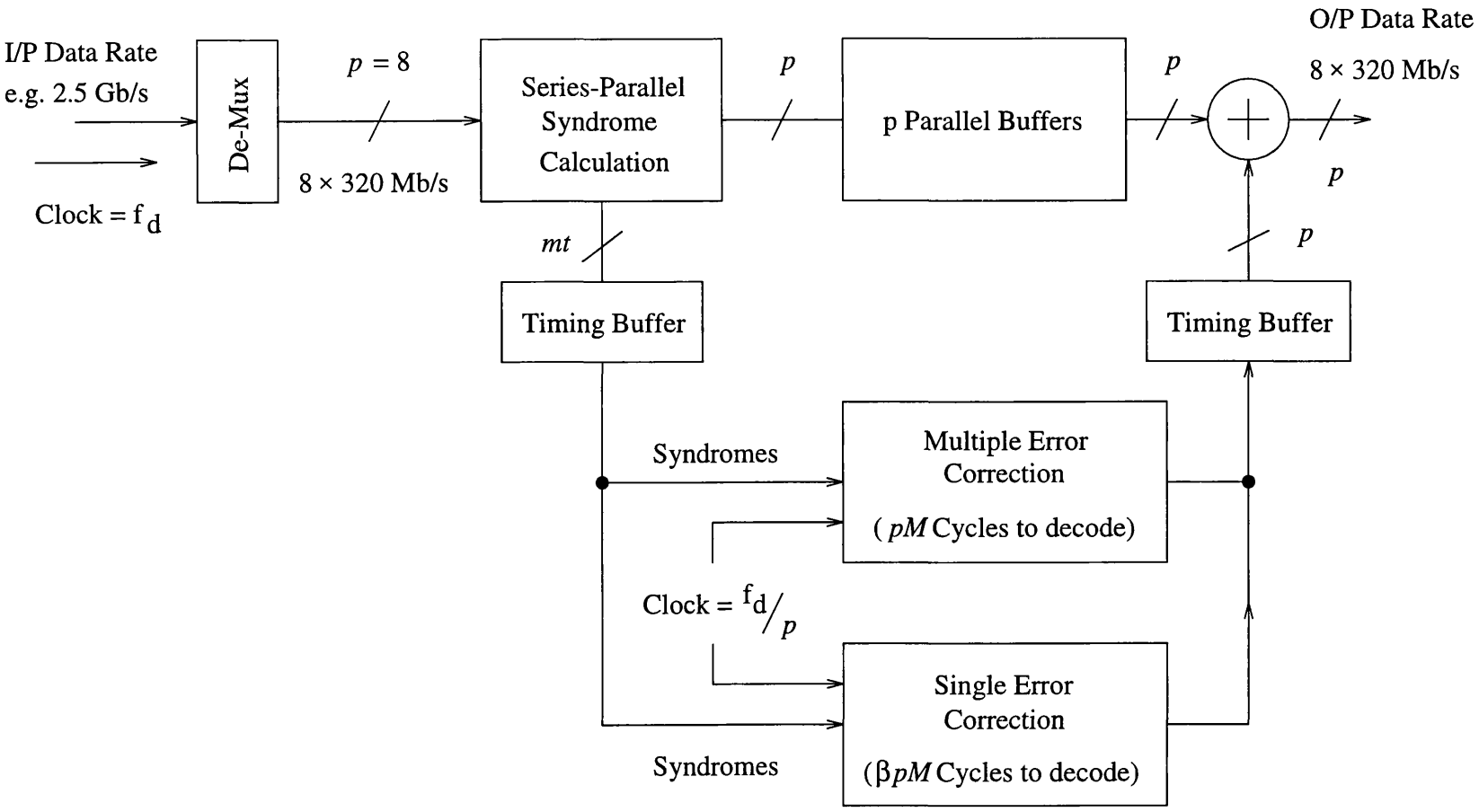


Figure 5.4: Optimised buffered decoding system.

5.5 Performance Impact

We shall consider several different high rate coding schemes with codeword lengths ranging from 127 to 1023 and error correcting capability from 4 to 6. A target output bit error rates of 10^{-10} will be assumed and a buffer lengths determined which result in $\text{Prob}\{BO(n)\} \leq 10^{-11}$. This is sufficient for the increase in residual error rate due to buffer overflow to be negligible.

Table 5.1 gives the parameters of the codes considered. It is clear that when $t \geq 5$ then an optimised decoding architecture is required if more than about an $\times 10$ increase in speed is desired. For the optimised decoder the figure quoted is a theoretical upper limit and is calculated for $\beta \to 0$.

(n, k, t)	λ	T	Potential Speed $\times 1/M$	
			worst case	optimised
(127,99,4)	0.044	127	22	1063
(127,92,5)	0.099	127	10	217
(127,85,6)	0.181	127	6	68
(255,223,4)	0.052	255	19	765
(255,215,5)	0.111	255	9	174
(255,207,6)	0.199	255	5	57
(511,475,4)	0.058	511	17	617
(511,466,5)	0.126	511	8	137
(511,457,6)	0.221	511	5	47
(1023,983,4)	0.068	1023	15	452
(1023,973,5)	0.139	1023	7	113
(1023,963,6)	0.243	1023	4	39

Table 5.1: Parameters of codes considered.

Figures 5.5-5.8 show, for an optimised decoder with a target output bit error rate of 10^{-10}, how the buffer lengths vary as M/T increases assuming a serial decoding process i.e. $T = n$ for each code considered above. In each case it is also assumed that the incoming data is de-multiplexed down by a factor of eight i.e $p = 8$ so that single error correction may be achieved in at most pT machine cycles.

In each of the figures the value quoted for the ratio M/T assumes that error correction is performed at an identical clock rate to that of the data. Therefore if $M/T = 80$ then for a decoder running at 2.5 Gb/s this figure indicates that multiple error correction must be performed in $80 \times T$ machine cycles at a clock rate of 2.5 GHz or $10 \times T$ machine cycles at 320 MHz when de-multiplexed down by a factor of eight.

To interpret the figures relating to buffer length for decoding in a series-parallel format, in this case p bits per clock cycle, then the total buffer length is exactly the same as in the figures, but split into p buffers of length B/p. For example if we consider the (255,223) t=4 code and assume M/T=100 then a buffer length of 4×10^5 bits is required for serial decoding but p buffers of length $4 \times 10^5/p$ bits are required if carried out in series-parallel form.

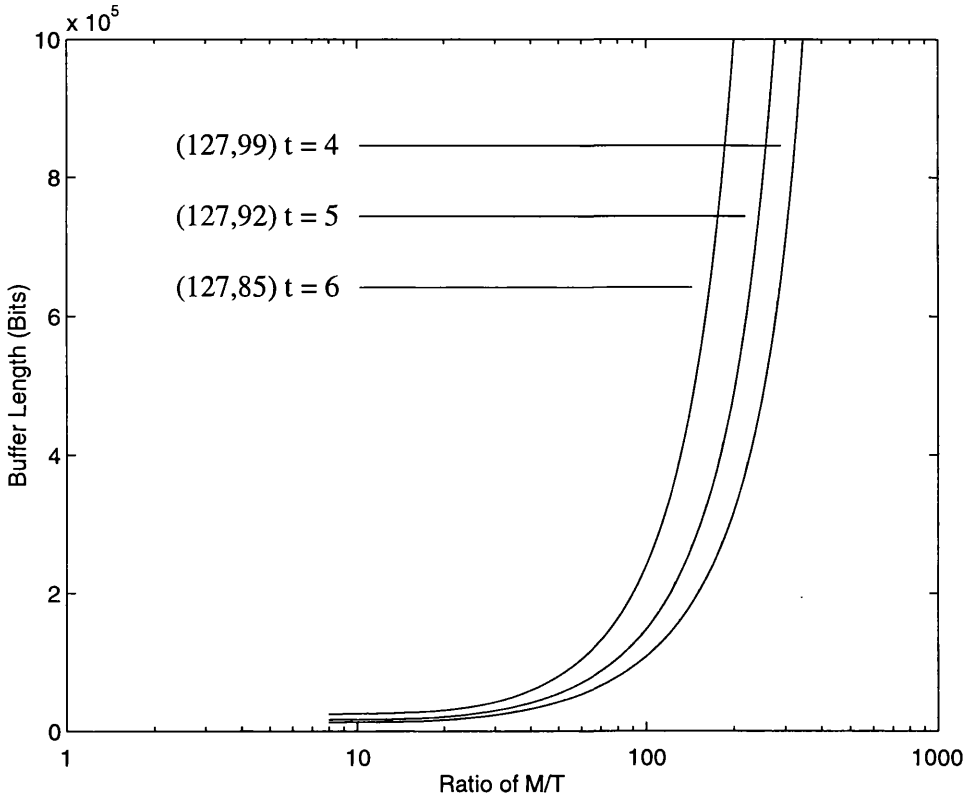


Figure 5.5: Buffer length vs M/T for $(127, k)$ codes.

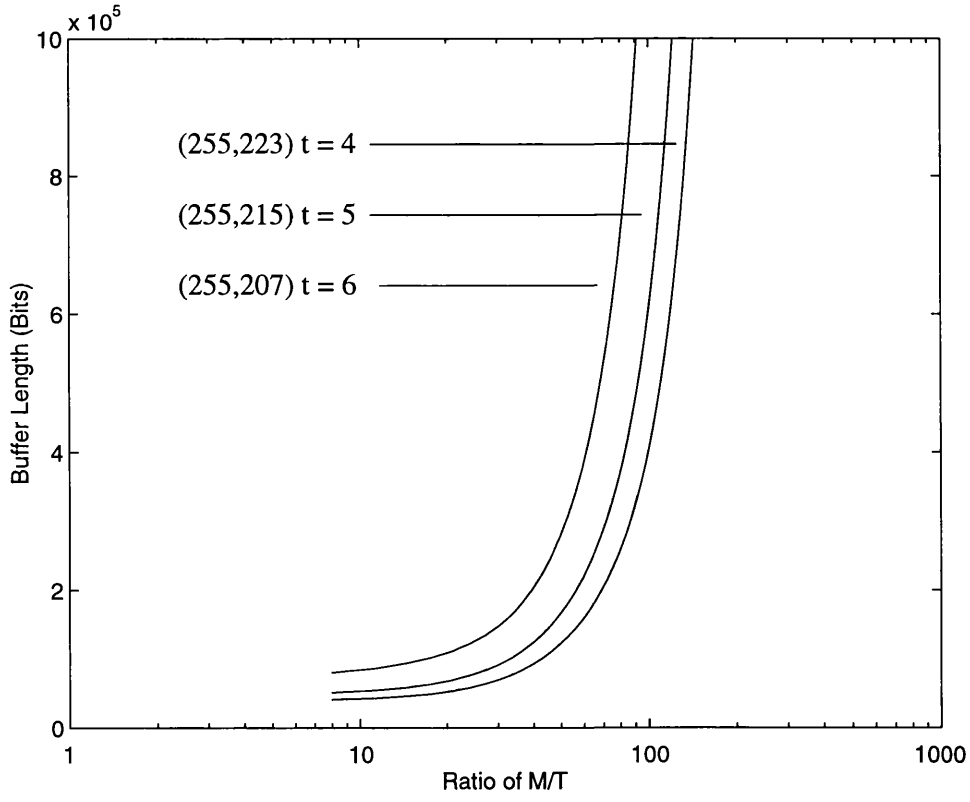


Figure 5.6: Buffer length vs M/T for $(225, k)$ codes.

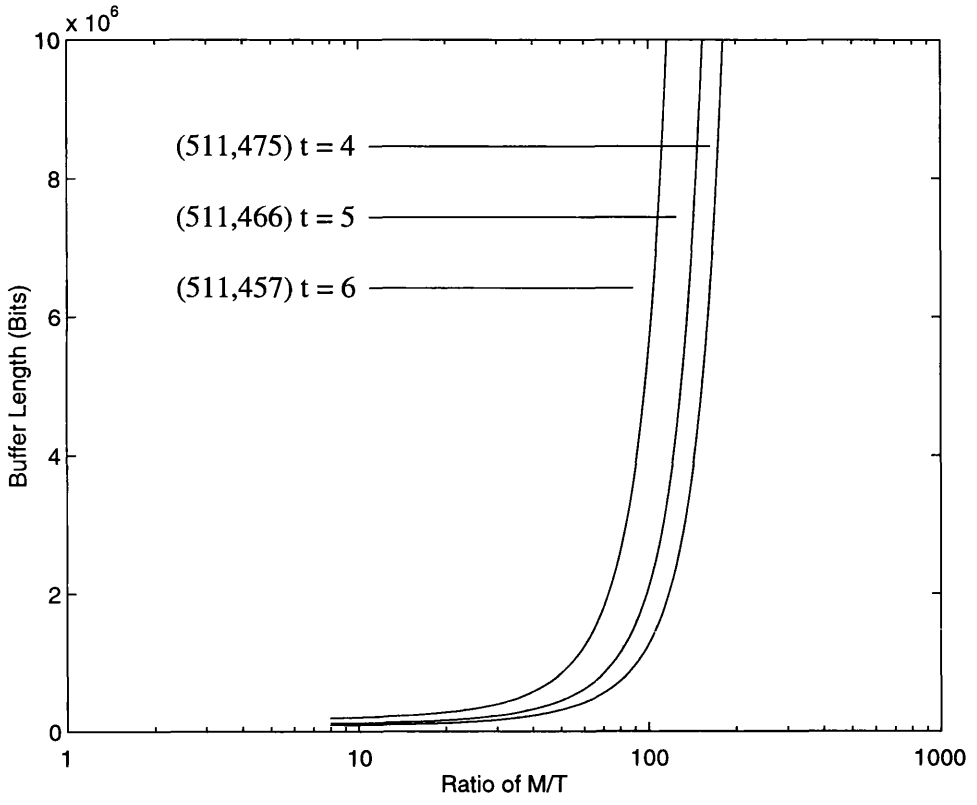


Figure 5.7: Buffer length vs M/T for $(511, k)$ codes.

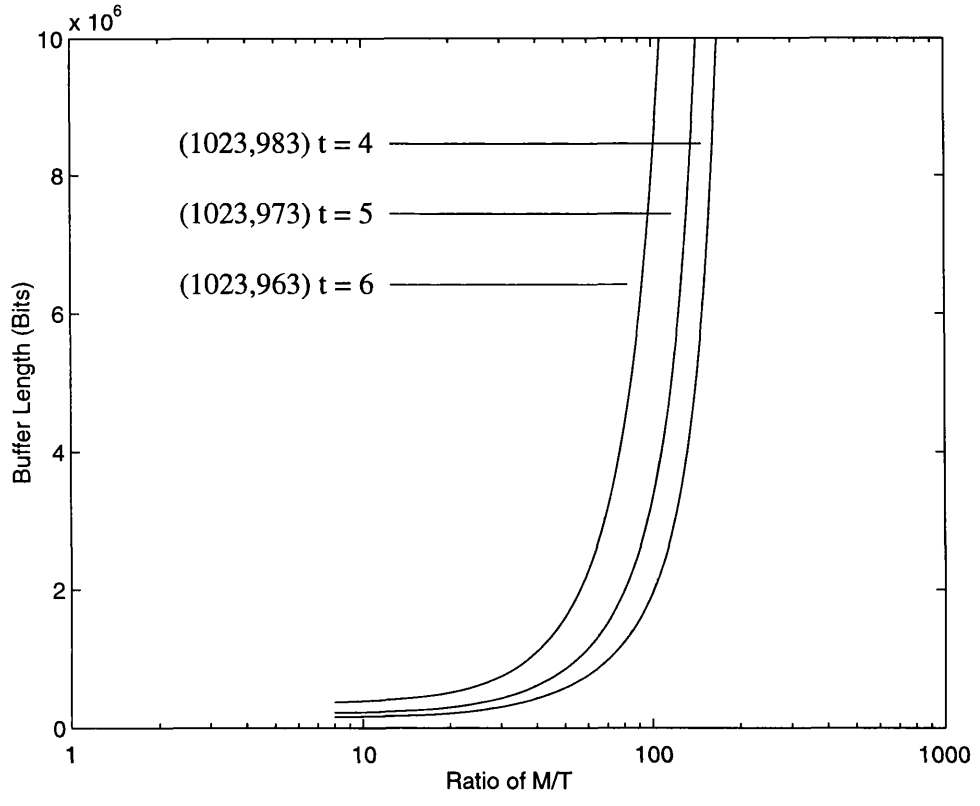


Figure 5.8: Buffer length vs M/T for $(1023, k)$ codes.

5.6 Buffered Decoding using a standard RS decoder

The concept of buffered decoding has been introduced in previous sections, in relation to binary BCH codes, as a means by which decoding speed may be significantly increased. Series-parallel syndrome calculation circuits, equivalent to those used for binary codes, are now employed to achieve high speed error detection for RS codes. Buffered decoding concepts are now applied to a decoding architecture which utilises a standard RS decoder IC as the main error correcting element. Analysis of such a decoder is undertaken to evaluate its performance in relation to high bit rate transmission systems. In particular, attention is focussed on a commercially available RS decoding IC, namely the L64710 RS codec produced by LSI Logic.

The L64710 codec is capable of encoding and decoding a (255,239) t=8 RS code with codeword symbols defined over GF(256). It is therefore capable of correcting 8 symbol errors per codeword and has good burst error correcting capability when used for binary transmission. Using a similar architecture to figure 5.1 a buffered decoder employing a standard RS decoding IC may be constructed as shown in figure 5.9. Unlike the binary decoder the series-parallel error detection circuit is used only to detect a non-zero syndrome, thus indicating the presence of errors. Once the syndrome has been evaluated only corrupt words are sent through the RS decoder.

By employing various algebraic techniques, we shall again assume that single errors may be treated as special cases. This leads to the realisation of an optimised decoder capable of operating at a further increased average speed. Using the analytical tools developed in earlier sections it is possible to establish an upper bound on the operational speed and buffer lengths required for various decoding scenarios.

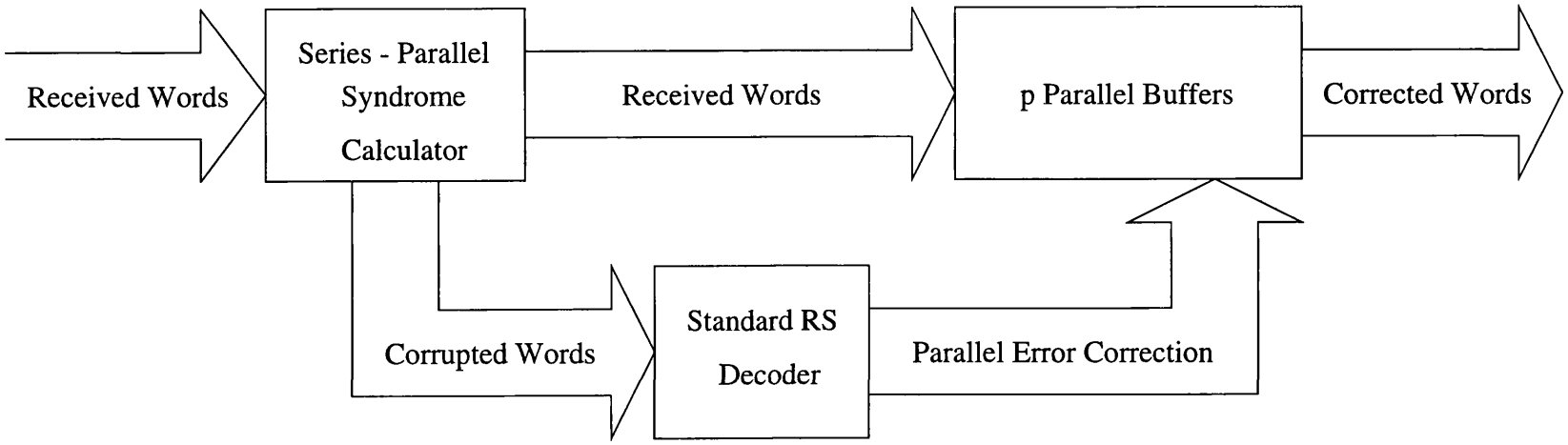


Figure 5.9: Block diagram of series-parallel buffered decoding system.

5.7 Approximations of Buffer Length and Decoding Speed

As with the binary decoder we shall consider two cases; (1) a worst case condition where it is assumed that all words in error take the same number of machine cycles to decode and (2) an optimised decoder. From the appropriate data sheet the following information relating to the codec was obtained.

n = 255 (i.e. $T = 255$) Machine cycles

λ = 0.495

M_i = 346 Machine cycles for all $i \geq 1$

Clock speed = 40 MHz

Where λ is the maximum allowable symbol error rate for the code to achieve an output BER $\leq 10^{-10}$. It is now possible to perform a numerical analysis on the buffered decoder based on the equations presented in sections 5.2 and 5.3.

5.7.1 Worst Case Decoder

Substituting the above values into equation (5.1) and assuming that $M_0 = 0$, an average decoding time of 135 machine cycles at 40 MHz is obtained for the worst case decoder. This value equates to a data rate of approximately 600 Mb/s. To achieve this, data must enter the delay line with a clock speed of 75 MHz. This dictates that one machine cycle must have a clock period of $(1/75) \times 10^{-6}$ seconds. With the decoding IC operating at the lower rate of 40 MHz, the delay line perceives the decoder as requiring 75/40×346 machine cycles (i.e. 648 machine cycles) to decode.

Substituting the above value into equation (5.13) gives

$$D^*(Z) - Z^T = \mathrm{e}^{-\lambda} + (1 - \mathrm{e}^{-\lambda})Z^{648} - Z^{255} = 0 \tag{5.18}$$

Solving for r_0 and substituting into equation (5.5) results in $r_0 = 1.0000399$ with a corresponding buffer length of 633 KBytes. The overall decoding delay resulting from the buffering process is approximately 8.4 ms.

5.7.2 Optimised Decoding

In order to increase the average decoding speed further, single error optimisation is investigated. Figure 5.10 illustrates the layout of an optimised decoder. On detection of a non-zero syndrome the corrupt codeword is sent through the RS decoder while the syndromes are simultaneously passed to the single error correction unit. Although both error correcting units operate in parallel optimised decoding will occur if the number of machine cycles required to correct single errors is less than that of multiple errors. Assuming that the error correcting capability of the code is not exceeded error correction will be achieved by the first unit to obtain a valid error vector.

Single error optimisation may be realised by employing similar methods to those used for binary BCH codes. However, when applied to RS codes the Meggitt decoder is required to store not one, but $2^q - 1$ syndromes in order to represent all possible non-zero error patterns which may occupy the highest order symbol position. A better solution, and one which avoids the need to store syndrome values, is to employ the Error Trapping technique. As in the binary case the Error Trapping decoder is only required to determine if the weight of the syndrome is t or less. For an (n, k) RS codeword single error correction can be achieved in n cycles at most.

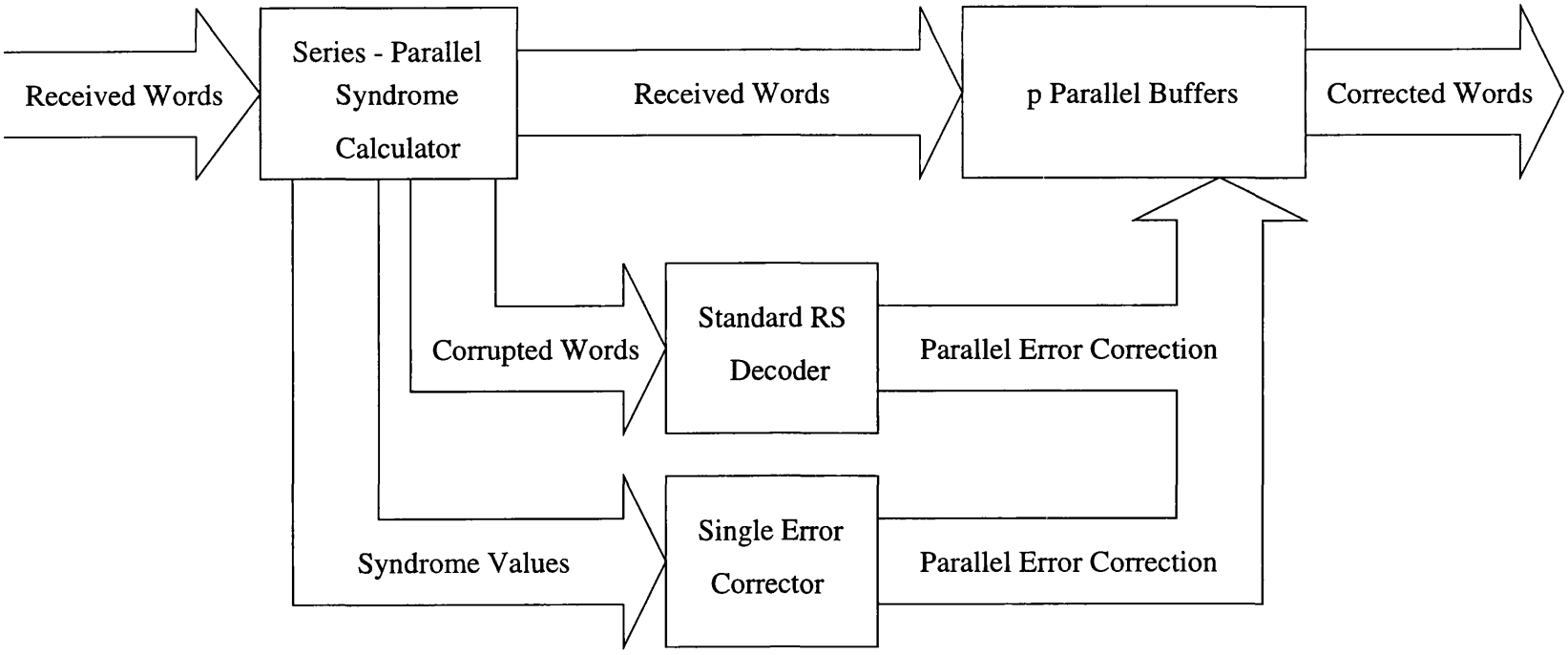


Figure 5.10: Optimised buffered decoding system.

Using equation (5.1) with β as a variable parameter it is possible to calculate the average decoding speed and hence the data rate of the buffered decoding system. Figure 5.11 illustrates the relationship between the achievable data rate and optimisation parameter β.

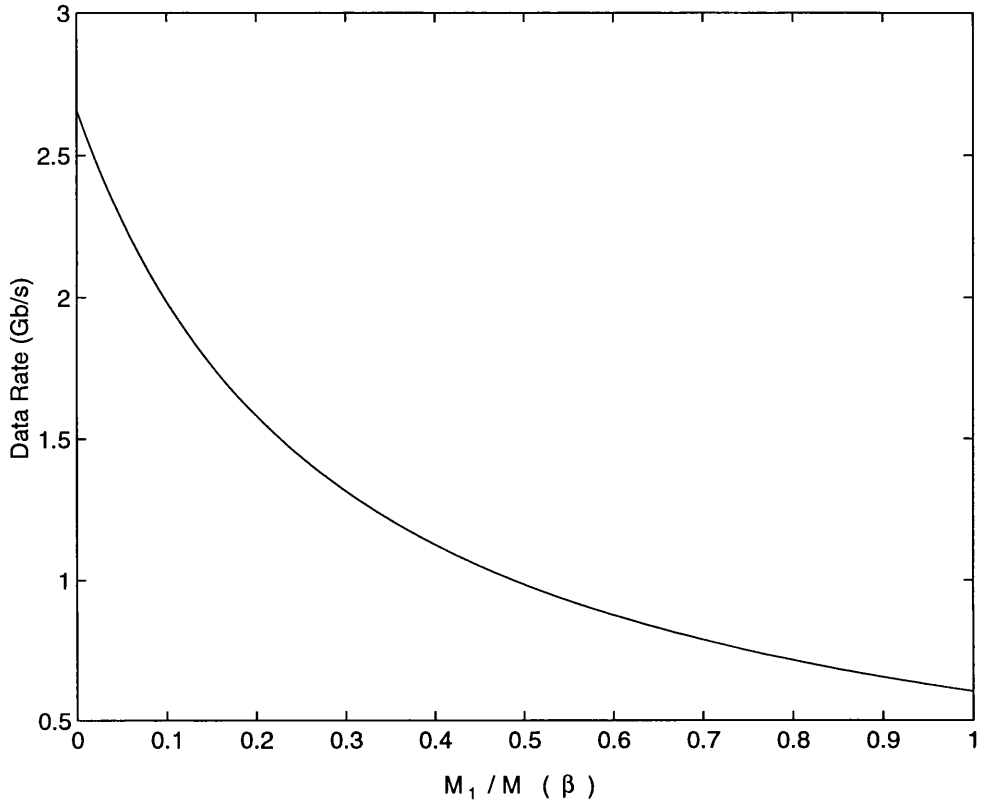


Figure 5.11: Data rate vs Optimisation Parameter (β).

Two significant values of note are $\beta = 1$ and $\beta = 0$. For $\beta = 1$, the worst case decoder, a data rate of 600 Mb/s is achievable, which is in agreement with the previous section. For $\beta = 0$, i.e. a best case decoder, the decoder may achieve a data rate of 2.6 Gb/s. This value indicates a theoretical upper bound placed on the data rate of the decoding system for single error optimisation.

For an optimised decoder using Error Trapping the value of β is found to be 0.73, this equates to an operational speed of 760 Mb/s. This figure represents more than a doubling in speed over the stand alone non-buffered decoding IC. To achieve this

operating speed a buffer length of a least 1.27 MBytes is required which results in an overall decoding delay of 13 ms.

To increase the data rate to 1Gb/s, for example, would require a β value of around 0.48. To achieve this the optimisation parameter must be reduced by a factor of 0.65. Since he decoding time for the IC is constant at 346 machine cycles, the number of cycles required to correct single errors must be reduced. In the case of the error trapping decoder the number of cycles required for single error optimisation is n. However we may effectively reduce this figure by operating the single error correction unit at an increased clock rate. If we consider applying a clock of 60 MHz to the single error corrector the number of cycles required to decode is now $40/60 \times n$ i.e $2/3$ n relative to the delay line. Based on our previous example we have effectively reduced β from 0.73 to 0.48. Higher data rates may be obtained in this way providing the clock rate does not exceed that supported by the circuitry responsible for single error correction.

5.8 Summary

In this chapter the concept and operation of a buffered decoder has been introduced and outlined. The use of series-parallel syndrome architectures for high speed error detection allows the decoding system to work at an average speed rather than its slowest speed. This work has been centred on the need for high speed decoders with operating speeds close to or at the line rate of high speed communications channels. This is of particular value as we have previously demonstrated encoding architectures operating at these rates.

With a prior knowledge of the channel statistics and by utilising the algebraic prop-

erties of BCH codes we then investigated the characteristics and functionality of an optimised decoder. Based on single error optimisation it was demonstrated that further increases in operational speed could be obtained with only modest amounts of memory.

The commercial availability of a standard RS decoding IC allowed a similar analysis to be performed on a non-binary decoder. Here it was shown that buffered decoding may be used to obtain an overall increase in decoding speed at the expense of increased decoding delay. Furthermore it was shown that a decoder operating at 1 Gb/s could be achieved if the time required for single error correction was compromised.

One important point of note is that during every analysis the value of λ was calculated to be the limiting value before the error correcting properties of each code were exceeded. Realistically, however, λ must be measured experimentally and it is this value which dictates the choice of coding scheme and residual bit error rate. Consequently the buffer lengths and decoding speeds presented here are viewed as being upper and lower limits respectively. When applied to high speed optical systems where relatively high signal to noise ratios are encountered it is non uncommon to find that λ can be orders of magnitude lower thus resulting in a substantial reduction in buffer length.

Chapter 6

FEC applications study: High speed optical systems

6.1 Introduction

Previous digital submarine cable systems have to date relied upon regenerative repeaters to boost signal levels where appropriate. Impairments such as intersymbol interference (ISI) and noise are isolated within each section and do not accumulate over the entire several thousands of kilometres of the system length involved in trans-oceanic links [48]. With the advent of the Erbium Doped Fibre Amplifier, (EDFA), system designers turned their attention to the applications of non-regenerative repeaters [49]. This was motivated by considering the prospect of reducing the complexity of the submerged units, enhancing the reliability, and maintaining optical transparency thereby allowing for possible future capacity upgrades by the enhancement of terminal equipment.

This approach though, whilst seductively simple in principle, brings with it many

potential pit-falls. In particular, transmission impairments which were previously of limited significance over the modest distances of a single repeater span can now accumulate to very significant levels over the long distances involved. To overcome these impairments without seriously impacting the system power budget, forward error control coding has been identified as one possible strategy for achieving error performance targets.

We begin this chapter by briefly highlighting the source of possible transmission impairments common to lightwave systems and discuss how their effects result in performance limitations. We then review the latest long haul optical networks which employ fibre amplifiers and asses how current and future systems will be effected by these impairments. Having introduced currently proposed coding methods for achieving target error rates we then suggest possible alternatives such as ARQ systems and low complexity binary codes.

6.2 Transmission Impairments

Transmission impairments in optical systems arise form many sources. Here we give a brief overview of some of the more frequently encountered causes of system degradation in unoptimised long span optical transmission systems. For a more rigorous analysis of the factors limiting the performance of lightwave systems the reader is referred to [50]. In the case of intensity modulated/direct detection systems, as found in the trans-atlantic and trans-pacific cable networks, the effects of phase fluctuations in the transmitted signal will not be discussed.

Generally we may group transmission impairments into the following classes: fibre and laser non-linearities, signal distortion due to single and multiple signals, and

and noise. For systems employing single mode lasers with external modulation i.e. operating in cw mode, the effects of laser chirp and relaxation can be ignored. However, the effects of fibre non-linearities manifest themselves in a number of ways. Stimulated Raman Scattering (SRS), Stimulated Brillouin Scattering (SBS) and Four Photon Mixing (FPM) are three examples of the effects caused by fibre non-linearities. The presence of SRS and SBS effectively limits the maximum power that can be launched into the fibre from either the transmitter or fibre amplifier. For single signal transmission SBS is often the dominating factor while for multi channel systems SRS becomes increasingly important as its effect is dependent on the number of channels propagating through the fibre. Four photon mixing occurs when the fibre non-linearities cause two co-propagating waves at different frequencies to mix and generate sidebands (this effect is similar to third order intermodulation products generated in RF mixers). These sidebands propagate with the initial waves and grow at their expense. The appearance of the additional waves as well as the depletion of the initial waves will degrade multichannel systems by crosstalk or excess attenuation [51].

Signal distortion (with a single signal) refers to impairments that distort the signal by broadening the width of the transmitted pulse [52]. Chromatic dispersion causes light of varying wavelengths to propagate through the fibre with different group velocities. This leads to a spreading of the received pulse thereby limiting the maximum bit rate due to ISI. Another type of dispersion is caused by the two fundamental polarisation modes propagating through the fibre with different velocities. Termed Polarisation Mode Dispersion (PMD) this effect causes the transmitted pulse to be received as two individual pulses separated by a delay equal to the difference in transit times of the two polarisation states. Again this leads to ISI limitations as pulses start to overlap in extreme cases. Although chromatic dispersion and PMD represent the main causes of signal distortion it is also known that further polar-

isation effects such as Polarisation Dependant Loss (PDL) and Polarisation Hole Burning (PHB) also impact on system performance [53].

The third class of transmission impairment for consideration is noise. Varying randomly from bit to bit noise in the received signal consists of shot noise, thermal noise, and, with the introduction of optical amplifiers, Amplified Spontaneous Emission (ASE). Shot noise is a result of quantum noise in the photodetector due to the fact that the received signal is a series of photons. The number of photons received during each symbol interval has a Poisson distribution [52]. Consequently the received signal level varies randomly from symbol to symbol. In particular it has been found that APDs as used in the TAT-12 system by STC exhibit significant shot noise which is dependent on both signal level and multiplication factor [54].

Thermal noise is a result of the random motion of electrons in a conductor. Introduced by the receiver pre-amplifier, thermal noise is assumed to additive Gaussian noise. ASE is additive white Gaussian noise in the optical domain that is dependent on the gain of the amplifiers. Although assumed to be random, with direct detection the electrical signal contains a noise times signal component. The noise in the received signal is therefore signal-level-dependent.

Although it is acknowledged that many other sources of transmission impairment exist, we have only highlighted the major ones which are commonly encountered in modern systems. The extent to which each impairment effects an individual system varies depending on the system parameters and configuration. For example fibre type, transmitter and receiver design, modulation format and the number and type of amplifiers used all give rise to, and interact with, the various impairments to a greater or lesser degree.

6.3 Optically amplified submarine networks

Along with the transpacific cable TPC-5 the transatlantic cables TAT-12/13 represent the first in a new generation of long haul optically amplified submarine systems operating SDH standards. The TAT-12/13 network consists of a fully protected fibre ring network linking North America and Europe. The two transatlantic cables span distances of 5913 km (TAT-12) and 6321 km (TAT-13) with optical amplifiers spaced every 45 km. Contained within each cable are two fibre pairs, one fibre pair designated service and the other restoration. Each service fibre pair transports two optically multiplexed STM-16 (2.5 Gb/s) signals each of which is generated by electrically multiplexing sixteen STM-1s (155 Mb/s). The result is a 5 Gb/s bit rate per service fibre giving a system capacity of 10 Gb/s. A further 10 Gb/s of capacity is offered by the restoration fibre pair in the event of a cable fault [55]. Figure 6.1 shows a block diagram of the TAT-12 cable system which connects Lands End in the UK with Green Hill in the USA.

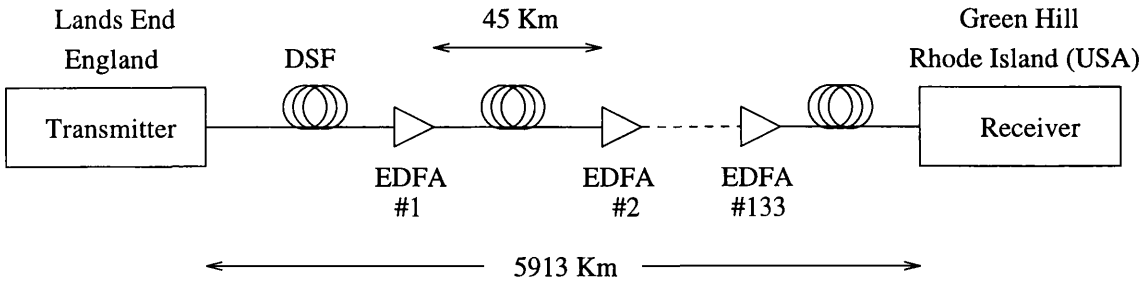


Figure 6.1: Block diagram of UK-US segment of the TAT-12 submarine cable.

Previous submarine systems such as TAT-(8-11) relied on complex electronic repeaters to amplify and re-time the signal electrically prior to optical re-transmission. Each repeater section was designed to operate error free under certain conditions such as wavelength, amplifier gain and data rate. As a consequence impairments and noise were confined to sections between repeaters and their effects need only

be considered over the modest distances involved. By integrating optical amplifiers into a system the signal remains in the optical domain throughout the entire transmission path. As a result the network is optically transparent and can accommodate future capacity upgrades. This may be achieved by an increase in bit rate or the addition of extra wavelength channels via wavelength division multiplexing (WDM) techniques as shown in figure 6.2.

Here several optical beams simultaneously propagate over a single fibre each with a different carrier wavelength $\lambda_1, \lambda_2, .., \lambda_n$ [56] [18]. An optical multiplexer couples the light from the various sources, in this case laser diodes (LDs), to the transmitting fibre. At the receiver an optical de-multiplexer separates the various carriers prior to photodetection. The capacity of such systems is often restricted by the limitations discussed in the previous section.

The use of Erbium doped fibre amplifiers (EDFAs) in lightwave systems has resulted in transmission impairments accumulating over the entire distance of the transmission path. It has been found that performance limitations, in the form of error floors, and system limitations such as transmitted/received power and fibre length are encountered due to the cumulative effects of these impairments [57]. Numerous experiments [58] [59] [60] have demonstrated that forward error control coding techniques can compensate for these effects or alternatively provide increased operating margins thereby allowing for less stringent operating parameters.

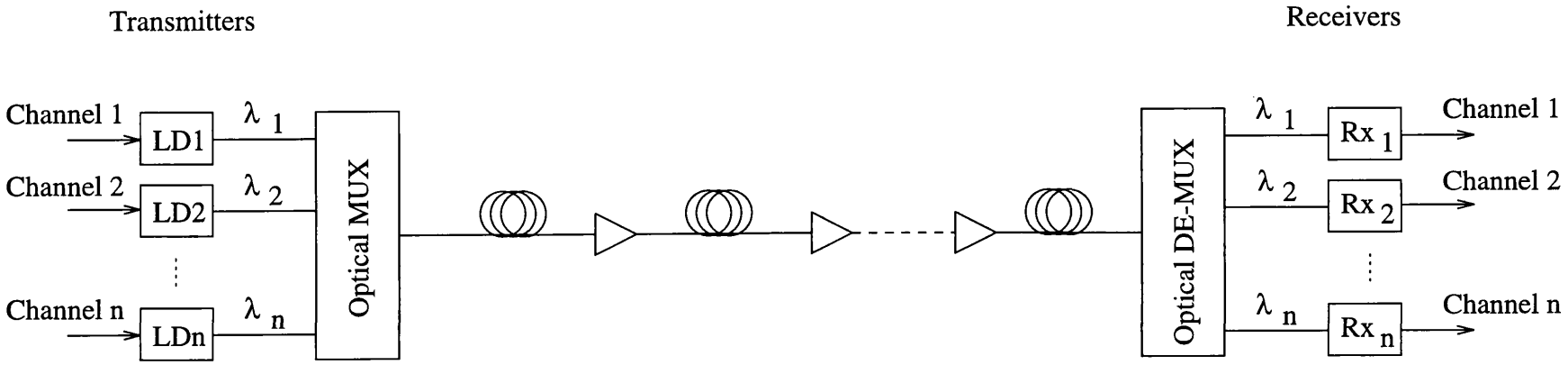


Figure 6.2: Block diagram of an optically amplified multichannel WDM system.

6.4 Current Proposal

Currently proposed methods, as shown in figure 6.3, involve the multiplexing of several tributary signals to produce the line rate signal. For submarine telecommunications line rates of 2.5 Gbit/s and 5 Gbit/s are of particular interest. At present error control coding units are not available at these rates and accordingly attention has been directed to the possibility of encoding each tributary data stream followed by multiplexing up to the line rate. Prompted significantly by ready availability of encoder and decoding integrated circuits operating at up to 320Mbit/s, times eight multiplexing has been proposed based upon each of the eight tributaries being encoded using a Reed-Solomon (255,239) code and then multiplexed to achieve a 2.5 Gbit/s transmission. At the receiver the incoming line signal is de-multiplexed and each of the individual tributaries decoded to enable error detection and correction.

It may be assumed that that errors occur independently and at random, this is due to the multiplexing operation which will, in the same way as interleaving, separate adjacent channel errors. The code performance can therefore be determined by the well-known error bound of equation 6.1 [20] with the results shown in figure 6.4.

$$P_{oe} \leq \frac{2^{q-1}}{2^q - 1} \sum_{i=t+1}^{n} \frac{i+t}{n} \binom{n}{i} P_{sym}^{i} (1 - P_{sym})^{n-i} \tag{6.1}$$

Where:

n = Codeword length.

q = No. of bits per codeword symbol (Binary codes q=1).

t = No. of correctable errors.

P_{oe} = Bit error rate at decoder output.

P_{ce} = Channel bit error rate.

P_{sym} = 1-$(1 - P_{ce})^q$.

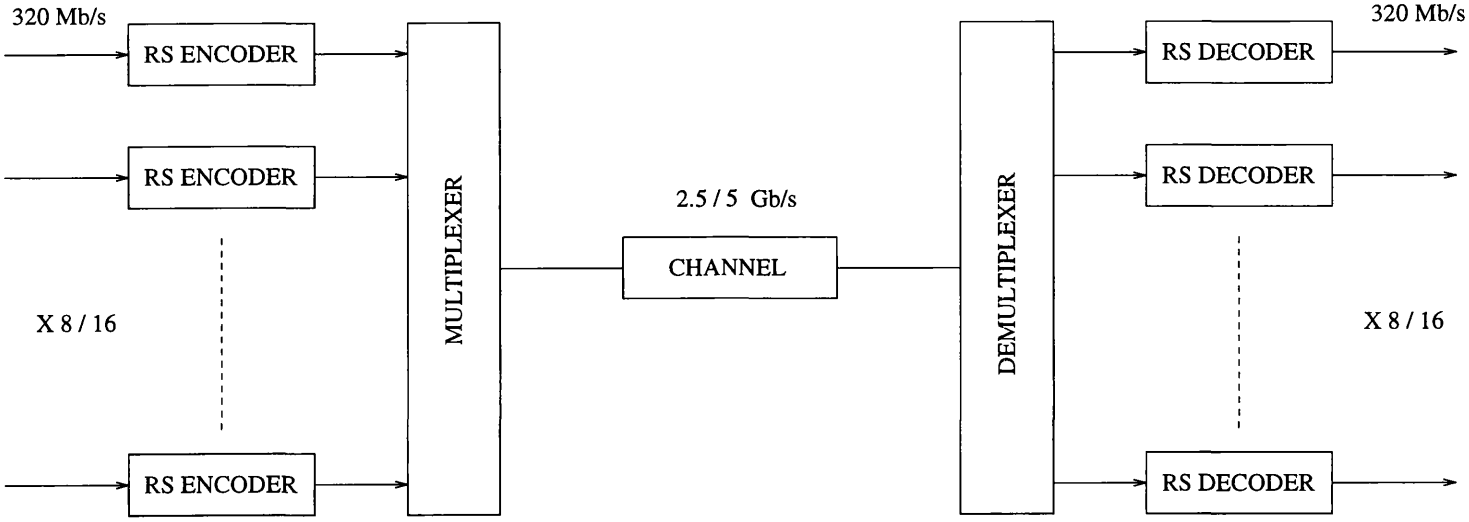


Figure 6.3: Proposed tributary encoding/decoding architecture.

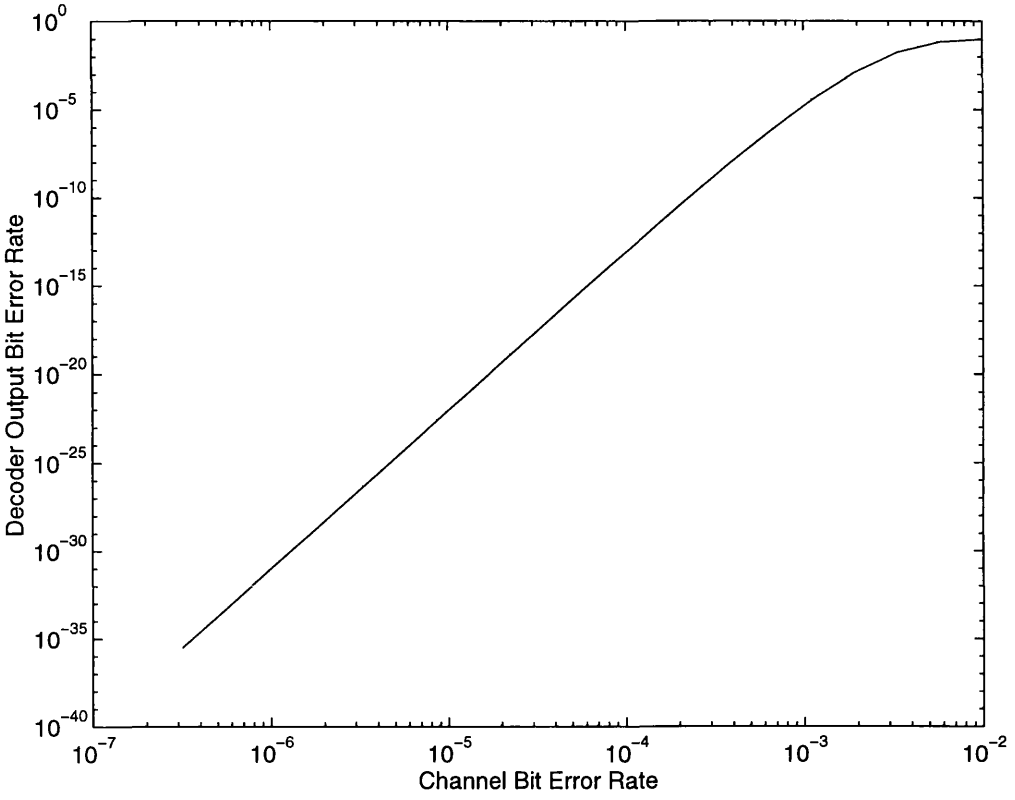


Figure 6.4: Error performance of Reed-Solomon (255,239) code.

6.5 System Requirement

For a typical telecommunications channel the decoded or residual bit error rate is of the order of 1 in 10^9 or better. In the case of the TAT-12/13 network the ITU Recommendation G.826 for error performance specifies a bit error rate better than 4×10^{-10} at the STM-1 level. The RS (255,239) code achieves this at a channel bit error rate in the region of 1 in 10^4. For lower channel error rates the performance improvement afforded by this code is much greater, with the residual error rate being approximately 1 in 10^{30} for a channel rate of 1 in 10^9 for example. It should be noted though that this is not a formal system requirement.

The tributary coding scheme requires multiple encoding and decoding circuits thereby making the implementation somewhat complex. This problem is compounded at higher transmission rates, for example a transmission rate of 5 Gbit/s would require sixteen encoders and decoders. Even then we have not taken into account future systems where WDM techniques are used to provide multiple channels. Current investigations have demonstrated system capacities of 20 $\times$ 5Gb/s channels which can only achieve error targets using FEC [61]. In this example the result is an extremely complex system requiring 320 encoders/decoders.

6.6 ARQ Systems

An alternative approach to achieving error performance targets is to use a form of coding known as Automatic Repeat Request or ARQ and it has been suggested [62] that this may be worthy of consideration in the present context. As stated in Chapter 2 the decoder is only required to detect the presence of errors and request only the re-transmission of any erroneous data blocks. As a result a feedback path must be available which is often inconvenient and increases system cost. The necessary request for re-transmission and the re-transmission itself obviously incurs a delay. Consider, for example, the TAT-12/13 transatlantic cable network which has a system length of approximately 6000 Km. Assuming the decoding time is negligible in comparison to the transmission time and the group velocity of light in the fibre is around $2 \times 10^8 \text{ms}^{-1}$. Then a 60 ms round-trip delay is experienced when a corrupt codeword is encountered. To accommodate for this delay buffers of sufficient length must be provided at the receiver in order to deliver the codewords in the correct order.

Furthermore, to achieve good error performance a block code with good error de-

tecting capabilities is required. For the redundancy in the transmitted data to be kept to a minimum codes with large block lengths are needed. Unfortunately the use of long block lengths does not provide a solution since the probability that a block contains errors increases rapidly with block length [20].

In addition, as the channel error rate increases information must be re-transmitted more frequently. The effective data rate is reduced accordingly up to the point where the average number of errors per codeword is one. At this point every received word is, on average, in error and the decoder must continuously request re-transmission. In such circumstances the performance of ARQ becomes unsatisfactory.

In summary ARQ systems are considered inappropriate for use in high speed optical systems due, primarily, to their inferior performance compared to FEC and delays associated with re-transmission and buffering at the receiver. Since encoding and error detection is already required for ARQ systems it may be argued that the increase in performance justifies the additional complexity required for error correction.

6.7 Low Complexity BCH Codes

Having considered the option of ARQ systems we are now motivated to consider low complexity binary codes, in particular BCH codes, since a simpler code may be realised at the line rate, obviating the need for replication of encoder and decoder circuits [63]. Figure 6.5 shows the performance curves of residual error rate versus channel error rate for representative cases: codeword length varying from 255 to 1023 bits and error correcting power ranging from 3 to 6 bit errors per codeword. We note that if a (1023,963) code is employed then a residual error rate of 1 in 10^9 or better is achieved provided the channel error rate does not degrade much

below about 1 in 10^4. This code thus satisfies the operational requirements of the system, achieving comparable performance to the more complex Reed-Solomon code in the critical region of system operation. Interestingly when the channel bit error rate exceeds 7×10^{-4} the binary BCH codes offer improved performance over the Reed-Solomon code.

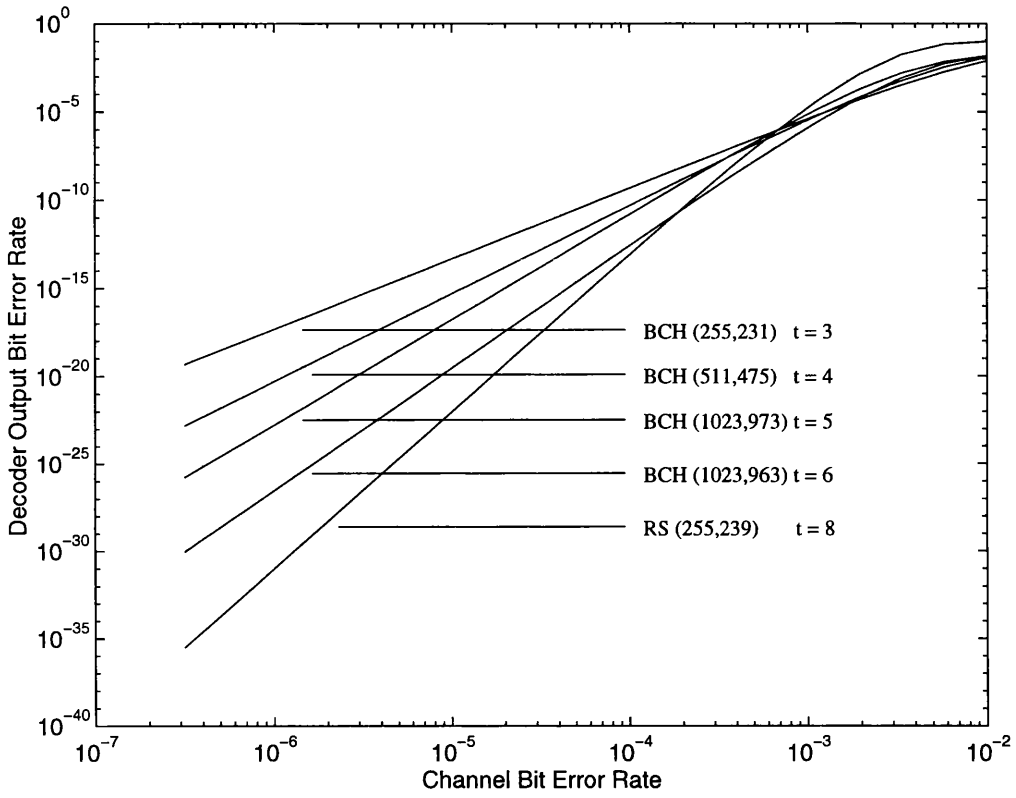


Figure 6.5: Performance comparison of BCH codes.

6.8 Summary

In summary we may conclude that FEC potentially provides the system designer with a method for achieving error performance targets in the presence of transmission impairments. The use of FEC allows for possible system upgrades or an increased operating margin for in-place systems. We have also demonstrated that

currently proposed methods, even on single channel systems, requires numerous encoding and decoding circuits. With the move to higher capacity systems this problem is further compounded. Motivated by the prospect of producing encoders and decoder operating at the line rate alternative methods such as low complexity binary BCH codes were investigated and shown to have comparable performance.

Chapter 7

Conclusions

This research has been primarily concerned with identifying and investigating novel generic architectures and approaches for the implementation of forward error control at very high bit rates. Previously reported methods for the generation and error detection of cyclic BCH codes were shown to have limitations in this context with respect to either circuit speed or complexity. Here series-parallel architectures have been identified as one possible strategy by providing an alternative approach which bridges the gap between the serial and parallel solutions.

Although applicable to encoding and error detection, series-parallel concepts are not readily adaptable to error correction. In this instance a technique referred to as buffered decoding has been demonstrated which utilises the error statistics associated with the communications channel to construct a decoder operating at an average rather than worst case speed. To conclude this research a case study detailing the implications of forward error control in current long haul optical channels was presented in order to provide a framework for this thesis.

7.1 Research Outcomes

The key outcomes of this research programme fall into three main areas.

1. Series-parallel architectures.
2. Buffered decoding.
3. The implications of forward error control coding in long haul optically amplified submarine systems.

Each of these will now be briefly considered.

7.1.1 Series-parallel architectures

Current methods for achieving error control coding at very high bit rates have, to date, relied on encoding several tributary data streams followed by multiplexing to the desired line rate, where the degree of multiplexing adopted determines the overall bit rate. At the receiver the data is de-multiplexed prior to decoding each individual tributary. In this instance replication of both encoding and decoding circuits is required and becomes prohibitively complex as the line rate increases.

An alternative approach is to produce a complete encoding/error detection structure which will encode or perform error detection on all tributary streams simultaneously. To achieve this a serial polynomial division circuit can be viewed as an autonomous system and mapped into a relevant transition matrix by way of its functional specification. Subsequent processing of this matrix yields intermediate circuit solutions with varying degrees of parallelism, thereby offering greater flexibility for the design of encoding and error detection systems.

The main achievements in this area may be summarised as follows:

- Investigation and analyses of current architectures employed for encoding and error detection.
- Proposed technique for generating series-parallel architectures based on the functional specification of polynomial division circuits for both binary and non-binary codes.
- Evaluation of speed and complexity improvements offered by series-parallel architectures.

7.1.2 Buffered decoding

By employing prior knowledge of channel error statistics a buffered decoding arrangement was presented. By performing error correction on only erroneous codewords a buffered decoder, which utilises high speed error detection, is shown to offer comparable performance to that of the encoding architectures previously presented. Using optimisation techniques to decode particular error patterns a further increase in operational speed may be achieved as well as reducing buffer lengths.

The main achievements in this area may be summarised as follows:

- Investigation of buffered decoding technique to achieve high speed error correction.
- Assessment of optimised decoder and in particular the effects of single error optimisation on buffer lengths and decoding speed.

- Assessment of buffered decoding technique when a standard Reed-Solomon decoding IC is employed as the error correcting unit.

7.1.3 FEC in long haul optically amplified systems

The use of erbium doped fibre amplifiers in a new generation of optical submarine systems has resulted in transmission impairments accumulating over the entire fibre span. The use of FEC in such systems has been identified as a possible solution for reducing these effects and achieving error performance targets. Some prospective coding systems are noted and their performance is assessed analytically. A coding scheme of much reduced complexity compared with currently proposed methods is identified and shown to offer comparable performance at the relevant channel error rates.

The main achievements in this area are:

- Assessment of impairments common to optically amplified lightwave transmission systems.
- Assessment of alternative techniques for reducing the effects of transmission impairments.
- Identification and performance comparison of low complexity binary codes with currently proposed multi-level RS codes for use in optical transmission systems.

7.2 Summary of Thesis

Chapter 2 introduced and reviewed several important concepts of forward error control coding. This was done with a view to providing an introduction to the notation used in subsequent chapters.

To supplement chapter 2 standard architectures for the generation and error detection of cyclic codes were introduced in chapter 3. In the case of serial encoding two different approaches based on the generator and parity polynomial of a code were considered for both binary and non-binary codes. To overcome the speed limitations of serial circuits, parallel architectures based on parity check equations were presented and shown to offer a substantial increase in operational speed at the expense of greatly increased circuit complexity. The chapter then concluded by discussing error detection methods. Again both serial and parallel forms were illustrated and shown to be subject to the similar speed/complexity constraints.

In order to provide a trade off between the high speed of parallel structures and the low complexity of serial circuits chapter 4 introduced series-parallel architectures. Using the concept of m-sequence generation as a foundation, series-parallel methods were shown to offer the prospect of increased operational speed without the demand for complex circuitry. Furthermore, by developing a process for generating such circuits, various structures could be defined thereby offering varying degrees of parallelism.

After presenting a solution to the problem of high speed error detection chapter 5 addressed the problem of high speed error correction. Unlike error detection error correction must be carried out in a serial manner, so an alternative technique to series-parallel methods was established. By buffering incoming data and using high

speed error detection a complete decoding system may be realised operating at an increased average speed. By performing a statistical analysis of the buffering arrangement it was possible to determine the length of the required buffers for various decoding scenarios. In addition, an investigation into the use of a commercially available decoding IC as the error correcting module was performed and shown to offer significant gains in terms of operational speed over stand alone devices.

Having demonstrated the basis for a complete FEC system, chapter 6 considered an applications case study involving the latest generation of long haul optically amplified networks. After discussing a number of known error inducing effects common to optical systems, currently proposed coding methods for such systems were reviewed. By considering the multiplexing operation required to achieve the desired line rate, reduced complexity binary codes were shown to offer comparable performance in the critical region of operation. In addition to this, binary codes have been shown to be well suited to the various encoding and decoding strategies presented in the previous chapters.

In summary this thesis has investigated novel generic architectures and techniques suitable for achieving forward error control at very high bit rates. These have included series-parallel structures applicable to both encoding and error detection circuits. In the case of error correction a buffered decoding arrangement has been proposed and shown to offer a significant gain in operational speed over standard error correction methods. The implications for systems applications have been discussed with particular emphasis on a new generation of long haul optically amplified lightwave systems.

7.3 Suggestions for Further Work

There are a number of areas of research resulting from this work that are worthy of further investigation. These are summarised as follows.

- A rigorous and thorough investigation of error statistics found in optically amplified systems to be validated by both experimental and simulation results.
- The implementation in hardware of series-parallel architectures based on the design procedure defined in this research.
- The design and assessment on new and efficient decoding algorithms for use in buffered decoding arrangements.
- Investigation of frequency domain encoding and decoding techniques which may offer the prospect of increased encoding and decoding speed and which may be directly realisable in the optical domain.
- The exploitation of coset codes to increase system performance by effectively limiting run-lengths.

In conclusion, this thesis has established and analysed novel architectures for implementing FEC at high bit rates. These provide a viable alternative to conventional encoding and decoding methods while offering superior performance in terms of speed. Moreover, it is envisaged that as future systems are deployed and present systems upgraded the use of FEC, particularly in high bit rate systems, will provide systems designers with a powerful tool for reducing the effects of transmission impairments.

Appendix A

Galois Field Arithmetic

This appendix introduces Galois Field arithmetic and examines several examples of circuit configurations for implementing arithmetic operations. Parameters relating to their complexity and computation time are provided.

Addition

Assume two field elements, β and γ, are to be added together. If β and γ are described in polynomial form as

$$\begin{aligned} \beta &= b_0 + b_1\alpha + b_2\alpha^2 + ... + b_{m-1}\alpha^{m-1} \\ \gamma &= c_0 + c_1\alpha + c_2\alpha^2 + ... + c_{m-1}\alpha^{m-1} \end{aligned}$$

then addition is achieved by the modulo-2 addition of the corresponding coefficients such that

$$\beta + \gamma = (b_0 + c_0) + (b_1 + c_1)\alpha + (b_2 + c_2)\alpha^2 + ... + (b_{m-1} + c_{m-1})\alpha^{m-1}.$$

For two m bit field elements this function may be realised by m EXOR-gates configured in the following parallel manner.

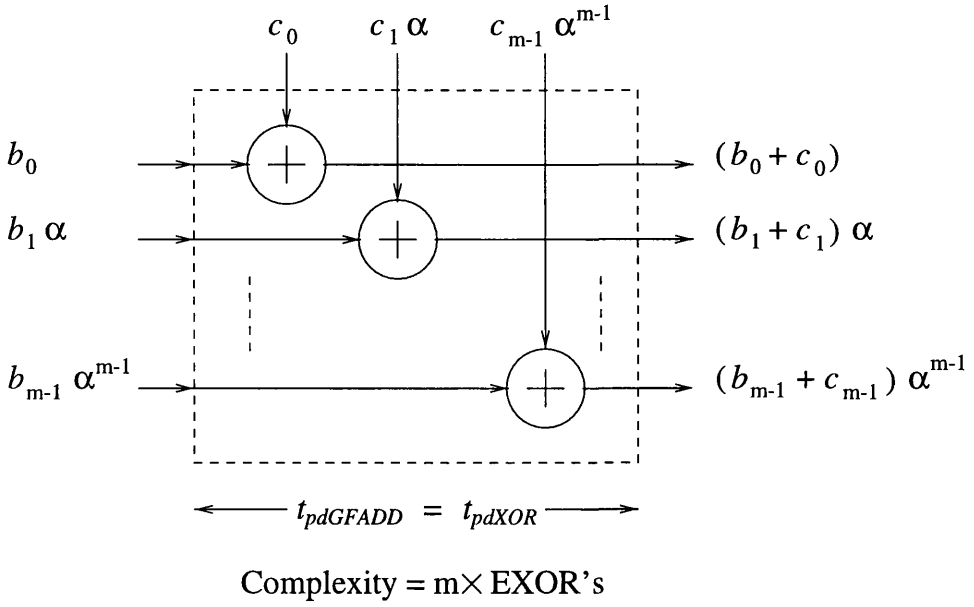


Figure A.1: Galois Field addition with symbols from GF(2^m).

Multiplication

To illustrate multiplication an example is given based on the Galois Field shown in Table A.1. For this example it shall be assumed that an arbitrary element β in GF(2^4) is to be multiplied by the element α^3. Again, if β is expressed in polynomial form,

$$\beta = b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3.$$

Multiplying both sides by of the equation by α^3 gives the following

$$\begin{aligned}\alpha^3\beta &= b_0\alpha^3 + b_1\alpha^4 + b_2\alpha^5 + b_3\alpha^6 \\ &= b_0\alpha^3 + b_1(1+\alpha) + b_2(\alpha+\alpha^2) + b_3(\alpha^2\alpha^3) \\ &= b_1 + (b_1+b_2)\alpha + (b_2+b_3)\alpha^2 + (b_0+b_3)\alpha^3.\end{aligned}$$

Based on the previous expression a circuit such as the one shown in figure A.2 can be formed.

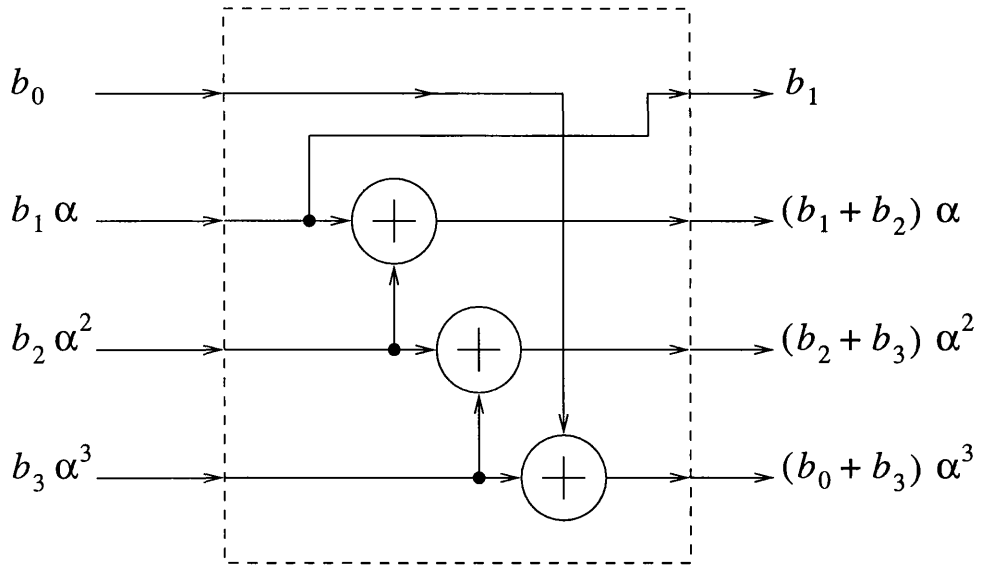


Figure A.2: Galois Field multiplier with symbols from GF(2^m).

The propagation delay of such a circuit is given by the longest path through the EXOR-gate network. In this example this is equal to the delay of a single EXOR-gate. However, in general one or more of the outputs may comprise of the modulo-2 addition of all the inputs. In this instance the delay is equal to the number of EXOR-gate stages. In the case of an 8-bit multiplier there is at most $\log_2(8) = 3$ stages. Hence the overall delay is equal to that of three EXOR-gates.

In terms of complexity, if each output contains all possible combinations of each input, then for an m bit multiplier $m - 1$ EXOR-gates are required for each output. Hence for the worst case where $m = 8$ the circuit contains $m \times (m - 1) = 56$ gates.

Power representation	Polynomial representation	4-Tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α^1	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1 + \alpha$	(1 1 0 0)
α^5	$\alpha + \alpha^2$	(0 1 1 0)
α^6	$\alpha^2 + \alpha^3$	(0 0 1 1)
α^7	$1 + \alpha + \alpha^3$	(1 1 0 1)
α^8	$1 + \alpha^2$	(1 0 1 0)
α^9	$\alpha + \alpha^3$	(0 1 0 1)
α^{10}	$1 + \alpha + \alpha^2$	(1 1 1 0)
α^{11}	$\alpha + \alpha^2 + \alpha^3$	(0 1 1 1)
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	(1 1 1 1)
α^{13}	$1 + \alpha^2 + \alpha^3$	(1 0 1 1)
α^{14}	$1 + \alpha^3$	(1 0 0 1)

Table A.1: Three representations for the elements of GF(2^4) generated by $p(X) = 1 + X + X^4$.

Appendix B

Introduction

In this appendix the Newton-Raphson method for locating the roots of a polynomial is presented. This method is found to be particularly efficient when dealing with polynomials of high degree such as those found in chapter 5.

B.1 Newton-Raphson Algorithm

To find the solution to the problem $f(x) = 0$ we perform the following.

1. Make an initial guess of the root defined by x_0
2. For n=0,1,2...., compute the following

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

where $f'(x_n)$ denotes the derivative of $f(x)$ with respect to x.

3. Compute x_1 from x_0, x_2 from x_1 and so on until the difference between x_{n+1} and x_n is less than some specified limit. The value of x_n is then the approximate solution to the equation $f(x) = 0$.

When programming this method for use on a computer it is often convenient to replace $f'(x_n)$ with

$$\frac{f(x_n + h) - f(x_n)}{h}$$

and use a sufficiently small value of h to achieve good accuracy [64]. In practice $h = 0.001$ proves to be a suitable value. To demonstrate this method consider equation 5.18 as given in chapter 5 with $\lambda = 0.495$.

$$D^*(Z) - Z^T = \mathrm{e}^{-\lambda} + (1 - \mathrm{e}^{-\lambda})Z^{648} - Z^{255} = 0 \tag{B.1}$$

Applying the Newton-Raphson method with $Z_0 = 1.1$ yields the following results.

$$\begin{aligned}
Z_0 &= 1.1 \\
Z_1 &= 1.098307456510710 \\
Z_2 &= 1.096617524963684 \\
&\vdots \quad \vdots \\
&\vdots \quad \vdots \\
Z_{2568} &= 1.000039991174907 \\
Z_{2569} &= 1.000039991174906 \\
Z_{2570} &= 1.000039991174906
\end{aligned}$$

With a specified accuracy of 10^{-15} it can be seen that $Z_{2569} = Z_{2570}$. Therefore the root of equation B.1 is found to be 1.000039991174906.

Bibliography

[1] P. COCHRANE. 'Future directions in long haul fibre optic systems', *British Telecom Technology Journal*, **8 No.2**: pp. 5–17, April 1990.

[2] R.D. SMITH. *Digital Transmission Systems.* Van Nostrand Reinhold, 1993.

[3] M. MATTHEWS and P. NEWCOMBE. 'The Synchronous Digital Hierarchy, Part 1', *IEE Review*, pp. 185–189, May 1991.

[4] M. MATTHEWS and P. NEWCOMBE. 'The Synchronous Digital Hierarchy, Part 2', *IEE Review*, pp. 229–233, May 1991.

[5] G.B. GABLA and J.L. CHABERT. 'Progress in repeaterless submarine systems'. Proceedings Suboptic, pp. 243–248, 1993.

[6] N.S. BERGANO *et al.* '100 Gb/s Error Free Transmission over 9100 km using Twenty 5 Gb/s WDM Data Channels'. OFC 96 Post Deadline Paper, pp. 23/1–23/5, February 1996.

[7] P.M. GABLA, J.L. PAMART, R. UHEL, E. LECLERC, J.O. FRORUD, F.X. OLLIVIER, and S. BORDERIEUX. '401 km, 622Mb/s and 357 km 2.488 Gb/s IM/DD repeaterless transmission experiments using erbium doped fibre amplifiers and error correcting code', *IEEE Photonics Technology Letters*, **4**: pp. 1148–1151, October 1992.

[8] A. ROBINSON. 'Experiences with forward error correction for optically amplified submarine systems'. IEE Colloquium on International Transmission Systems, pp. 9/1–9/6, February 1994.

[9] S. YAMAMOTO, H.TAKAHIRA, and M. TANAKA. '5 Gbit/s Optical Transmission Terminal Equipment using Forward Error Correcting Code and Optical Amplifier', *IEE Electronics Letters*, **30 No.3**: pp. 254–255, February 1994.

[10] J.L. PAMART *et al.* 'Forward Error Correction in a 5 Gbit/s 6400 km EDFA Based System', *IEE Electronics Letters*, **30 No.4**: pp. 342–343, February 1994.

[11] S. YAMAMOTO, H.TAKAHIRA, E. SHIBANO, M. TANAKA, and Y.C. CHEN. 'BER Performance Improvement by Forward Error Correcting Code in 5 Gbit/s 9000 km EDFA Transmission System', *IEE Electronics Letters*, **30 No.9**: pp. 718–719, April 1994.

[12] E.R. BERLEKAMP, R.E. PEILE, and S.P. POPE. 'The Application of Error Control to Communications', *IEEE Communications Magazine*, **25 No.4**: pp. 44–57, April 1987.

[13] C.E. SHANNON. 'A Mathematical Theory of Communication', *Bell Syst. Tech. J.*, **27**: pp. 379–423, July 1948.

[14] C.E. SHANNON. 'A Mathematical Theory of Communication', *Bell Syst. Tech. J.*, **27**: pp. 623–656, October 1948.

[15] R.C.V. MACARIO. *Modern personal Radio Systems.* The Institute of Electrical Engineers, 1996.

[16] A. HOCQUENGHEM. 'Codes corecturs d'erreurs', *Chiffres*, **2**: pp. 147–156, 1959.

[17] R.C. BOSE and D.K. RAY-CHAUDRY. 'On a Class of Error Correcting Binary Group Codes', *Inf. Control*, **3**: pp. 68–79, March 1960.

[18] D. GORENSTEIN and N. ZIERLER. 'A Class of Cyclic Linear Error Correcting Codes in p^m Symbols', **J**. *Soc. Ind. Appl. Math.*, **9**: pp. 107–214, June 1961.

[19] I.S. REED and G. SOLOMON. 'Polynomial Codes over Certain Finite Fields', **J**. *Soc. Ind. Appl. Math.*, **8**: pp. 300–304, June 1960.

[20] S. LIN and D.J. COSTELLO. *Error control coding: fundamentals and applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[21] J.L. MASSEY. 'Step-by-step Decoding of the Bose-Chaudhuri-Hocquenghem Codes', *IEEE Transactions on Information Theory*, **IT-11 No.4**: pp. 580–585, October 1965.

[22] W.W. PETERSON. 'Encoding and Error-correction Procedures for the Bose-Chaudhuri Codes', *IRE Transactions on Information Theory*, **IT-6**: pp. 459–470, September 1960.

[23] E.R. BERLEKAMP. 'On Decoding Binary Bose-Chaudhuri-Hocquenghem Codes', *IEEE Transactions on Information Theory*, **IT-11 No.4**: pp. 577–579, October 1965.

[24] E.R. BERLEKAMP. *Algebraic Coding Theory.* McGraw-Hill, New York, 1968.

[25] R.T. CHIEN. 'Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes', *IEEE Transactions on Information Theory*, **IT-6**: pp. 357–363, October 1964.

[26] J.L. MASSEY. 'Shift Register Synthesis and BCH Decoding', *IEEE Transactions on Information Theory*, **IT-15 No.1**: pp. 122–127, January 1969.

[27] R.E. BLAHUT. *Theory and practice of error control codes.* Addison-Wesley, 1983.

[28] G. LONGO. *Algebraic Coding and Applications.* Wein-New York.

[29] LSI Logic. *Reed-Solomon Encoders/Decoders Technical Manual.* LSI Logic Corporation, 1993.

[30] J.A. COEKIN and J.R WICKING. 'Generating and correlating pseudo-random binary sequences at 1 gigabit/second'. Conference on Digital Instrumentation (*IEE* Conference Proceedings No. 106), pp. 139–144, November 1973.

[31] O'REILLY J.J. 'Series-parallel generation of m-sequences', *The Radio and Electronic Engineer*, **45**: pp. 171–176, April 1975.

[32] GRUBER J., MARTEN P., PETSCHACHER R., and RUSSER P. 'Electronic circuits for high bit rate digital fiber optic communication systems', *IEEE Transactions on Communications*, **COM-26**: pp. 1088–1098, July 1977.

[33] HUGHES J.B., COUGHLIN J.B., HARBOTT R.G., VAN DEN HURK T.H.J., and VAN DEN BERGH B.J. 'A versatile ECL multiplexer IC for the Gbit/s range', *IEEE Journal of Solid-State Circuits*, **SC-14**: pp. 812–817, Oct 1979.

[34] B. HOLDSWORTH. *Digital Logic Design.* Butterworths, 1988.

[35] D.A. HODGES nad H.G. JACKSON. *Analysis and Design of Digital Integrated Circuits.* McGraw Hill, 1988.

[36] W.R. BLOOD. *MECL System Design Handbook.* Motorola Semiconductor Products, 1988.

[37] R.S. TUCKER, E. EISENSTEIN, and S.K. KOROTKY. 'Optical Time-Division Multiplexing for Very High Bit Rate Transmission', *IEEE Journal of Lightwave Technology*, **6 No.11**: pp. 1737–1749, November 1988.

[38] M.J. O'MAHONY. 'Optical Multiplexing in Fiber Networks: Progress in WDM and OTDM', *IEEE Communications Magazine*, pp. 82–88, December 1995.

[39] E.R. BERLEKAMP and R.J. McELIECE. *The impact of processing techniques on communications*, pp. 145–158. Martinus Nijhoff, 1985.

[40] L. KLIENROCK. *Queueing Systems, Vols. 1 and 2.* Wiley, New York, 1975.

[41] P.V. O'NEIL. *Advanced Engineering Mathematics.* Wadsworth, 1987.

[42] J.E. MEGGITT. 'Error correcting codes and their implementation', *IRE Transactions on Information Theory*, **IT-7**: pp. 232–244, October 1961.

[43] T. KASAMI. 'A decoding procedure for multiple error correcting cyclic codes', *IEEE Transactions on Information Theory*, **IT-10**: pp. 134–139, April 1964.

[44] A. BENYAMIN-SEEYAR, S.G.S SHIVA, and V.K BHARGAVA. 'Capability of the error-trapping technique in decoding cyclic codes', *IEEE Transactions on Information Theory*, **IT-32**: pp. 166–180, March 1986.

[45] S. WEI and C. WEI. 'High-speed Hardware Decoder For Double-error-correcting Binary BCH Codes'. volume 136 Pt. I No.3, pp. 227–231. *IEE Proceedings*, June 1989.

[46] S. WEI and C. WEI. 'A High Speed Real-Time Binary BCH Coder', *IEEE Transactions on Circuits and Systems For Video Technology*, **3 No.2**: pp. 138–147, April 1993.

[47] E. BUSHEHRI, V. BRATOV, A. THIEDE, V. STAROSELSKY, and D. CLARK. 'Design and analysis of a low power HEMT SRAM cell', *IEE Electronics Letters*, **31 No.21**: pp. 1828–1829, October 1995.

[48] J.J. O'REILLY, A. POPPLEWELL, and R. BLAKE. 'FEC for Future Trans-Oceanic Optical Systems'. pp. 78–82. Fifth *IEE* Conference on Telecommunications, Brighton, March 1995.

[49] N.A. OLSSON. 'Lightwave Systems with Optical Amplifiers', *IEEE Journal of Lightwave Technology*, **7 No.7**: pp. 1071–1081, July 1989.

[50] D. MARCUSE, A.R. CHRAPLYVY, and R.W. TKACH. 'Effect of fiber nonlinearity on long distance transmission', *IEEE Journal of Lightwave Technology*, **9 No.1**: pp. 121–128, January 1991.

[51] A.R. CHRAPLYVY. 'Limitations on lightwave communications imposed by optical fiber nonlinearities', *IEEE Journal of Lightwave Technology*, **8 No.10**: pp. 1548–1557, October 1990.

[52] J.H. WINTER, R.D. GITLIN, and S. KASTURIA. 'Reducing the effects of transmission impairments in digital fiber optic systems', *IEEE Communications Magazine*, pp. 68–76, June 1993.

[53] A. LORD. 'Polarisation effects in TAT 12'. IEE Colloquium on Transoceanic cable communications - TAT 12 & 13 herald a new era, pp. 6/1–6/5, March 1996.

[54] D.J.H. MACLEAN. *Optical line systems.* Wiley, England, 1996.

[55] G.L. MARLE, M. COLAS, M. GREEN, I.D. ANDREWS, and G. REA. 'TAT 12/13 Cable Network - The need for the Ring'. IEE Colloquium on Transoceanic cable communications - TAT 12 & 13 herald a new era, pp. 1/1–1/8, March 1996.

[56] J.V. PALAIS. *Fiber optic communications.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[57] S. SAITO, T.IMAI, and T. ITO. 'An over 2200-km coherent transmission experiment at 2.5 Gb/s using Erbium-Doped Fiber In-Line amplifiers', *IEEE Journal of Lightwave Technology*, **9 No.2**: pp. 161–167, February 1991.

[58] P.M. GABLA *et al.* '401 km, 622 Mb/s and 357, 2.488 Gb/s IM/DD Repeaterless Transmission Experiments Using Erbium-Doped Fiber Amplifiers and Error Correcting Code', *IEEE Photonics Technology Letters*, **4 No.10**: pp. 1148–1151, October 1992.

[59] W.D. GROVER. 'Forward Error Control in Dispersion Limited Lightwave Systems', *IEEE Journal of Lightwave Technology*, **6 No.5**: pp. 643–653, July 1988.

[60] J-H. WU and J. WU. 'Performance of Reed-Solomon Codes in CPFSK Coherent Optical Communications', *Journal of Optical Communications*, **13 No.1**: pp. 19–22, July 1992.

[61] J-B. THOMINE, G. AUBIN, and F. PIRIO. 'Future trends for high capacity optically amplified submarine systems'. IEE Colloquium on Transoceanic cable communications - TAT 12 & 13 herald a new era, pp. 8/1–8/6, March 1996.

[62] P.M. Lane. *Private Communication.* 1996.

[63] J.J. O'REILLY, A. POPPLEWELL, and R. BLAKE. 'Forward error control for international telecommunications transmission'. IEE Colloquium on International Transmission Systems, pp. 8/1–8/4, February 1994.

[64] W.H. PRESS, S.A. TEUKOLOSKY, W.T. VETTERLING, and B.P FLANNERY. *Numerical Recipes in C.* Cambridge University Press, 1992.