ISBN 0-315-55498-3

THE UNIVERSITY OF ALBERTA

Guided Scrambling: A New Line Coding Technique For Optical Fiber Communication Systems

by

© Ivan J. Fair

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

Department of Electrical Engineering

Edmonton, Alberta

Fall, 1989

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR:	Ivan J. Fair
TITLE OF THESIS:	Guided Scrambling: A New Line Coding Technique for Optical Fiber Communication Systems
DEGREE:	Master of Science
YEAR GRANTED:	1989

[illegible]

13707 - 109 Avenue
Edmonton, Alberta
T5M 2G8

Date: July 24/89

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Guided Scrambling: A New Line Coding Technique For Optical Fiber Communication Systems** submitted by **Ivan J. Fair** in partial fulfillment of the requirements for the degree of **Master of Science**.

[illegible]

[illegible]

Supervisors

[illegible]

[illegible]

Date: July 24/89

ABSTRACT

This thesis introduces a new line coding technique for communication systems which transmit two-level signals. Due to its similarity to the established technique of self-synchronizing scrambling and its additional ability to guide the scrambling process to produce a balanced encoded bit stream, this technique is called Guided Scrambling. Because it can be implemented with simple combinational and sequential logic circuits, this technique is a candidate for use in high bit rate transmission systems such as light-intensity modulated optical fiber systems.

The thesis begins with an overview of current line coding techniques and their applicability to high bit rate systems. The concept of Guided Scrambling is then presented, followed by derivation of code parameters which ensure good line code characteristics. The performance of a number of Guided Scrambling configurations is reported in terms of maximum consecutive like encoded bits, encoded stream disparity, decoder error extension, and power spectral density of the encoded signal. Simulation results agree with theoretical expectations. Comparison of Guided Scrambling with conventional line coding techniques indicates performance which is superior to current non-alphabetic line codes and not much inferior to alphabetic techniques.

ACKNOWLEDGMENTS

I would like to thank the following people and institutions for their contributions to this work:

- Mr. W.D. Grover, who introduced me to the study of transmission coding,

- Drs. R.I. MacDonald and W.A. Krzymien, for supervising this project,

- the members of the examining committee, for taking the time to review this work,

- Mr. D.M. Johnson, for technical assistance during creation of the Pull-Down Menu Shell,

- Mr. T.D. Bilodeau, who provided the program upon which line code spectral analysis in this thesis is based,

- the Natural Sciences and Engineering Research Council, Bell Northern Research Ltd., the Alberta Telecommunications Research Centre, and the University of Alberta for financial assistance,

- the Alberta Telecommunications Research Centre and its staff and students, for providing an environment in which this work could be efficiently carried out,

- my family, who understood when I couldn't always make it over for Sunday dinner,

and, most importantly, Ms. Jane P. Leske, without whose patience, support, and encouragement this work would never have been completed. Let's get married, babe.

CONTENTS

Chapter 1 Introduction 1

1.1 Thesis Overview 1

1.2 Notation 2

1.2.1 Polynomial Representation of a Bit Stream 2

1.2.2 Guided Scrambling Notation 3

Chapter 2 Line Coding for Optical Fiber Transmission Systems 5

2.1 Code Radix 5

2.2 Line Code Requirements 6

2.3 Line Codes for Fiber Systems 7

2.3.1 Unbalanced Codes 7

2.3.1 Balanced Codes 11

Chapter 3 The Concept of Guided Scrambling 14

3.1 Mathematics of Self-synchronizing Scrambling 14

3.2 Basis of Guided Scrambling 15

Chapter 4 Guided Scrambling Configurations 19

4.1 Placement of Augmenting Bits 20

4.2 Potential Scrambling Polynomials 21

4.2.1 Quotient Set Relationships 21

4.2.2 Scrambling polynomial for high encoded stream transition density, $n_a = 1$ 22

4.2.3 Scrambling polynomials for balanced transmission, $n_a = 1$ 23

4.2.3 Potential scrambling polynomials for multiple bit augmentation 24

4.3 Quotient Selection Criteria 26

4.4 Shift Register Update 28

Chapter 5 Guided Scrambling Performance 32

5.1 Code Performance 32

5.2 Spectra of Encoded Signals 40

5.3 Line Code Simulation 48

5.3.1 Line Coding Environment 49

5.3.2 Extent of Simulation 49

5.3.3 Encoded Stream WRDS and Digital Sum Variation 50

5.3.4 Consecutive Like Bit Distribution 53

5.3.5 Encoded Stream Transition Density 57
5.3.6 Error Extension 60
5.4 Performance Summary 62

Chapter 6 Conclusion 64
6.1 Thesis Summary 64
6.2 Further Work 65

References 67

Appendix 1 Notes on Guided Scrambling 69

Appendix 2 Guided Scrambling Performance Analysis 75

Appendix 3 Guided Scrambling Spectral Analysis 115

Appendix 4 Computer Simulation Environment 122

Appendix 5 Glossary 140

LIST OF TABLES

Table 2.1 Primitive Irreducible Polynomials 8

Table 3.1 Modulo-2 Arithmetic Operations 14

Table 4.1 $r_i(x)$ for $n = 8$, $n_e = 2$, $d(x)$ $x^2 + 1$ 22

Table 4.2 Minimum Weight Scrambling Polynomials for Balanced Guided Scrambling, $n_e = 1$ 24

Table 4.3 Bit Sequence Representation of Scrambling Polynomials associated with $r(x) = 1011 \dots 11$ 24

Table 4.4 Quotient Relationships. $n = 10$, $n_e = 2$, $d(x) = x^2 + 1$ 25

Table 4.5 Quotient Relationships. $n = 10$, $n_e = 3$, $d(x) = x^3 + 1$ 25

Table 4.6 Quotient Relationships. $n = 10$, $n_e = 4$, $d(x) = x^4 + 1$ 25

Table 5.1 Guided Scrambling Performance Summary: $n_e = 1$, $d(x) = x^2 + 1$ 33

Table 5.2 Guided Scrambling Performance Summary: $n_e = 1$, $d(x) = x + 1$ 34

Table 5.3 Guided Scrambling Performance Summary: $n_e = 1$, $d(x) \in D_{max}(x)$ 35

Table 5.4 Guided Scrambling Performance Summary: $n_e = 1$, $d(x) = x^{n-2} + x^{n-3} + 1$ 36

Table 5.5 Guided Scrambling Performance Summary: $n_e = 1$, $d(x) = x^{n-1} + x^{n-2} + 1$ 37

Table 5.6 Guided Scrambling Performance Summary: $n_e = 2$, $d(x) = x^2 + 1$ 38

Table 5.7 DSV and L_{max} for the Alphabetic 7B8B and Partially Flipped family of block codes 40

Table 5.8 Guided Scrambling Configurations Simulated 50

Table 5.9 Representative Guided Scrambling DSV Simulation Results 53

Table 5.10 Average Simulated Error Extension 62

Table A1.1 Relationship between #ones and #zeros in $q_0(x)$ and $q_1(x)$ 74

Table A2.1 Minimum Encoded Stream Transition Density. $n_e = 1$, $d(x) = x^2 + 1$ 77

Table A2.2 Maximum Quotient Tail Lengths. $n_e = 1$, $d(x) = x^2 + 1$ 81

Table A2.3 Quotients which exhibit $T_{A,max}$. $n_e = 1$, $d(x) = x^2 + 1$ 82

Table A2.4 Maximum Quotient Head Lengths. $n_e = 1$, $d(x) = x^2 + 1$ 83

Table A2.5 Quotients which exhibit $H_{B,max}$. $n_e = 1$, $d(x) = x^2 + 1$ 84

Table A2.6 Guided Scrambling Performance Summary: $n_s = 1, d(x) = x^2 + 1$ 96

Table A2.6 Guided Scrambling Performance Summary: $n_s = 1, d(x) = x + 1$ 94

Table A2.8 Bit Sequence Representation of Polynomials from family $d(x) \in D_{[illegible]}(x)$ 95

Table A2.9 Guided Scrambling Performance Summary: $n_s = 1, d(x) \in D_{[illegible]}(x)$ 99

Table A2.10 Guided Scrambling Performance Summary: $n_s = 1, d(x) = x^{n-2} + x^{n-3} + 1$ 102

Table A2.11 Guided Scrambling Performance Summary: $n_s = 1, d(x) = x^{n-1} + x^{n-2} + 1$ 102

Table A2.12 Situations requiring further quotient selection. $n_s = 2, d(x) = x^2 + 1$ 103

Table A2.13 Number of transitions within quotients for each quotient selection set configuration. $n_s = 2, d(x) = x^2 + 1$ 105

Table A2.14 Selected Quotient $|WD|_{max}$. $n_s = 2, d(x) = x^2 + 1$ 106

Table A2.15 Quotient Selection Sets from which quotients exhibiting $|WD|_{max}$ must be selected. $n_s = 2, d(x) = x^2 + 1$ 106

Table A2.16 $T_{max}, H_{max}, L_{max}$ when n mod 4 = 0, $n_s = 2, d(x) = x^2 + 1$ 110

Table A2.17 $T_{max}, H_{max}, L_{max}$ when n mod 4 = 1, $n_s = 2, d(x) = x^2 + 1$ 110

Table A2.18 $T_{max}, H_{max}, L_{max}$ when n mod 4 = 2, $n_s = 2, d(x) = x^2 + 1$ 111

Table A2.19 $T_{max}, H_{max}, L_{max}$ when n mod 4 = 3, $n_s = 2, d(x) = x^2 + 1$ 111

Table A2.20 Summary of shortest L_{max} and their decision mechanisms. $n_s = 2, d(x) = x^2 + 1$ 112

Table A2.21 Guided Scrambling Performance Summary: $n_s = 2, d(x) = x^2 + 1$ 114

Table A3.1 Block GS7B8B Encoder State Assignment 117

Table A3.2 Transmitted Quotient and Encoder Next State for all zero input word 117

Table A4.1 Definition file Colour Specification 123

Table A4.2 Definition file Prompt Specification 124

Table A4.3 Windowed DOS commands 127

Table A4.4 Bit sequence history information retained in .BIT files 130

Table A4.5 Line Coding Environment files 131

LIST OF FIGURES

Figure 2.1 Block Diagram of a Digital Communications System 5

Figure 2.2 Shift Register Generation of a Pseudo-random Bit Sequence 9

Figure 2.3 Reset Scrambling with 7 stage PRBS generation 9

Figure 2.4 Shift Register Polynomial Division, with and without delay 10

Figure 2.5 Shift Register Polynomial Multiplication 10

Figure 2.6 Alphabetic Code Implementation 12

Figure 3.1 Polynomial Division 16

Figure 3.2 Polynomial Multiplication 17

Figure 3.3 Conceptual diagram of Guided Scrambling 18

Figure 4.1 Efficient Implementation of Guided Scrambling 19

Figure 4.2 Generation of the alternating sequence 23

Figure 4.3 Block Guided Scrambling, $n_a = 1$ 29

Figure 4.4 Continuous Guided Scrambling, $n_a = 1$ 30

Figure 5.1 Comparison of Maximum Consecutive Like Bits vs. Digital Sum Variation 39

Figure 5.2 Power Spectral Density: Alphabetic 7B8B and PF7B8B Codes 42

Figure 5.3 GS7B8B Power Spectral Density: $d(x) = x + 1$ 43

Figure 5.4 GS7B8B Power Spectral Density: $d(x) = x^6 + x^4 + x^3 + x + 1$ 43

Figure 5.5 GS7B8B Power Spectral Density: $d(x) = x^6 + x^5 + 1$ 45

Figure 5.6 GS7B8B Power Spectral Density: $d(x) = x^7 + x^6 + 1$ 46

Figure 5.7 GS14B16B Power Spectral Density: $d(x) = x^2 + 1$ 47

Figure 5.8 Fundamental Frequency of the Alternating Sequence 47

Figure 5.9 Representative Guided Scrambling Encoded Stream WRDS Distributions 51

Figure 5.10 Conventional Line Code Encoded Stream WRDS Distributions 52

Figure 5.11 Conventional Line Code Consecutive Like Bit Distributions 54

Figure 5.12 Representative Guided Scrambling Consecutive Like Bit Distributions 55

Figure 5.13 Encoded Stream Transition Density vs. Source Stream Logic One Probability 58

Figure 5.14 Simulated Error Extension 61

Figure A1.1 Relationships between transitions in $q_0(x)$ and $q_1(x)$ 72

Figure A2.1 Situation for maximum consecutive like bits. $n_s = 1, d(x) = x^2 + 1$ 77

Figure A2.2 Actual vs. Approximate Decoded Error Rate. Continuous Decoding, $n_s = 1, d(x) = x^2 + 1$ 85

Figure A2.3 Situation for maximum consecutive like bits. $n_s = 1, d(x) = x + 1$ 89

Figure A2.4 Situation for maximum consecutive like bits. $n_s = 2, d(x) = x^2 + 1$ 109

Figure A3.1 Two Part Continuous GS Encoder Model, $n_s = 1$ 118

Figure A3.2 Formation of $a_0''(x)$ 119

Figure A4.1 Line Coding Environment Display 131

Figure A4.2 Line Coding Environment Pull-Down Menus 132

CHAPTER 1
INTRODUCTION

Optical Fiber transmission systems operating today transmit binary, light-intensity modulated bit sequences. To allow for simplified receiver and clock recovery design, the signals they transmit should contain a large number of transitions and a small low frequency content [1-2]. The transformation of a source bit sequence into a sequence containing these characteristics and the subsequent recovery of the original signal is called Line Coding.

Line coding has been the object of much study [1-17]. In particular, many line codes have been developed for fiber transmission systems due to their range of applications and varying requirements.

This thesis introduces a new line code developed specifically for high bit rate fiber systems. Guided Scrambling is a balanced, non-alphabetic line coding technique which exhibits low error extension and can be configured to yield balanced transmission with a high encoded stream transition density. Its simple encoding and decoding procedures can be implemented with high speed digital circuitry, making it a good candidate for implementation in high speed fiber systems.

Following a brief overview of the thesis, this chapter introduces notation which will be used throughout code development and analysis.

1.1 Thesis Overview

In a more complete introduction to the field of line coding, Chapter 2 details line code requirements and surveys the techniques presently used in optical fiber transmission systems. Particular emphasis is placed on the applicability of these line coding techniques to high bit rate applications. Following a new interpretation of self-synchronizing scrambling processes, Chapter 3 reveals a property of this established line coding technique which can be exploited to improve transmitted stream characteristics. In doing so, the concept of Guided Scrambling is introduced.

Chapter 4 derives Guided Scrambling code parameters which ensure good line code characteristics. The performance of a number of GS configurations is reported in Chapter 5. Tables of values for maximum consecutive like encoded bits, encoded stream digital sum variation, error extension and minimum encoded stream transition density are compared with corresponding values for conventional line coding techniques. The effect that Guided Scrambling has on the power spectral density of the transmitted bit stream is reported, and simulation results which agree with theoretical expectations portray encoded stream characteristics which vary with source stream logic one probability.

Chapter 6 concludes the thesis by summarizing the concept, configurations, and performance of Guided Scrambling and offering suggestions for continued analysis of this coding technique.

1.2 Notation

The explanation and analysis of Guided Scrambling is simplified when standard notation is introduced for bit stream representation and measures of line code performance. In Guided Scrambling encoding and decoding procedures, bit streams are best represented as polynomials. Section 1.2.1 describes this polynomial representation and defines standard notation for commonly occurring bit streams. Section 1.2.2 presents further notation used in the description and analysis of Guided Scrambling codes.

1.2.1 Polynomial Representation of a Bit Stream

Polynomials provide a convenient form for bit stream representation. If the value of the most significant or first bit in time is represented by a coefficient associated with the variable of highest degree, the value of the second bit is represented by the coefficient of second highest degree and so on until the least signifi^ant bit or last bit is represented by the least significant coefficient, a bit stream can be represented by the polynomial:

$$b(x) = c_{j-1}x^{j-1} + c_{j-2}x^{j-2} + c_{j-3}x^{j-3} + \ldots + c_1x + c_0 , \tag{1.1}$$

where the bit sequence is of length j and the values of the coefficients c_i are 0 or 1. For example, the following bit sequence representations are equivalent:

$$\text{bit stream} = 10010100011 ,$$
$$b(x) = x^{10} + x^7 + x^5 + x + 1 .$$

Since these notations are equivalent, it is often convenient to combine them in the following shorthand notation:

$$b(x) = 10010100011 .$$

Contrary to usual notation, this representation places the first bit in time on the left rather than on the right.

Standard notation for commonly occurring bit sequences has been adopted during the development of GS codes. The importance of the following sequences will become apparent during code development.

$d(x)$ = scrambling polynomial, of degree d.

$s(x)$ = source bit stream.

$a_i(x)$ = augmented source word. The subscript i denotes the decimal representation of the binary value of the augmenting bits. For example, a source word augmented with the bits 01 would be denoted $a_1(x)$ while a word augmented with the bits 11 would be denoted $a_3(x)$.

$q_i(x)$ = quotient which results from the division of $a_i(x)$ by $d(x)$.

$r_i(x)$	=	the bit sequence which relates $q_i(x)$ to $q_0(x)$ through modulo-2 addition. When the only quotients considered are $q_1(x)$ and $q_0(x)$, the subscript from $r_1(x)$ will be omitted.
$rm(x)$	=	remainder of the division of $a_i(x)$ by $d(x)$.
$tx(x)$	=	transmitted bit stream.
$e(x)$	=	error pattern introduced during transmission.
$rx(x)$	=	received bit stream.
$u(x)$	=	decoded bit stream.

1.2.2 Guided Scrambling Notation

The following abbreviations, symbols, and operators will be used during the development and analysis of Guided Scrambling:

m	=	length of source word.
n	=	length of encoded word.
n_a	=	number of bits which augment each source word.
p	=	the probability of a logic one in the source bit sequence. Alternatively, this can be interpreted as the normalized source stream dc level.
RDS	=	Running Digital Sum: the accumulated sum of bit sequence digit values from an arbitrary time origin. Contrary to the conventional digit value assignment of +0.5 for a mark and -0.5 for a space [20], analysis in this thesis has been simplified with the assignment of the digit values +1 for a mark and -1 for a space. RDS_{max} and RDS_{min} represent maximum and minimum values of stream RDS respectively.
WRDS	=	Word-end Running Digital Sum: RDS sampled at the end of a word. Maximum and minimum values are denoted $WRDS_{max}$ and $WRDS_{min}$ respectively.
WD	=	Word Disparity: the digital sum difference over a word. Maximum and minimum values are denoted WD_{max} and WD_{min} respectively.
DSV	=	Digital Sum Variation: usually defined as the difference between maximum and minimum values of RDS. To allow for comparison with previously published values, the DSV for bit sequences considered in this thesis will be calculated as:

$$DSV = \frac{1}{2}(RDS_{max} - RDS_{min}) . \tag{1.2}$$

c	=	the center or midpoint of digital sum excursion. Its value is given by:

$$c = \frac{1}{2}(RDS_{max} + RDS_{min}) . \tag{1.3}$$

When subscripted, it also denotes a coefficient of a polynomial as given in Equation (1.1).

to_i	=	the transition opportunity preceding the bit of degree i. An opportunity for a transition is assumed to exist prior to the first bit of a bit stream and between every pair of subsequent adjacent bits.
t	=	the number of transitions associated with a word, including the transition which might precede its most significant bit.
t_{int}	=	the number of transitions which occur within a word. This excludes the transition which might occur immediately preceding the word.
td	=	transition density: the number of transitions in a bit sequence divided by the number of transition opportunities. It has units of [transitions/bit] and is limited in value to 0.0 - 1.0.
L_{max}	=	the maximum number of consecutive like bits in a sequence.
Q	=	the operator which denotes formation of a quotient sequence through modulo-2 division of its argument by its subscripted polynomial.

CHAPTER 2
LINE CODING FOR OPTICAL FIBER TRANSMISSION SYSTEMS

Line coding denotes the modification of a source symbol sequence to allow for proper signal reception in the presence of transmission impairments. More precisely, line coding refers to the encoding and decoding procedures which perform this transformation and its inverse. Figure 2.1 indicates the placement of line encoder and decoder in a digital communications system.

This chapter begins with a discussion of the reasons for past and continued use of two-level signalling in fiber transmission systems, and then outlines the required characteristics of these signals. The chapter concludes with an overview of current line coding techniques which attempt to provide such signalling, with an emphasis on the applicability of these line codes to high bit rate transmission systems.

2.1 Code Radix

With few exceptions, light-intensity modulated fiber transmission systems use binary signalling. In the past, this choice for two level coding was dictated by the non-linearity of lasers and avalanche photodiodes (APDs) and the associated circuit complexity multi-level signals require. However, recent advances in laser technology have increased laser linearity and the shift towards transmission at longer wavelengths has resulted in wider use of the PIN diode receiver, a device which allows for uniform spacing of signal and decision levels. But even with these technical advances and the corresponding reduction in practical disadvantages for multi-level coding, it has been shown that multi-level transmission in fiber systems offers little or no advantage over binary transmission [2]. Even in situations where a slight performance advantage might be realized, the additional complexity required for multi-level transmitter and receiver design cannot yet be justified. Accordingly, only two-level signalling will be considered here.

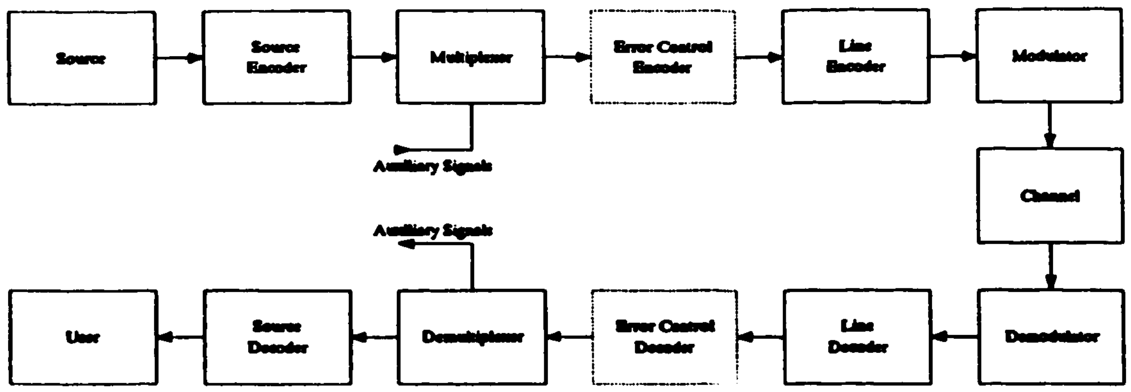


Figure 2.1: Block diagram of a Digital Communications System

2.2 Line Code Requirements

In order that the optical receiver can correctly receive and decode the transmitted signal, this signal must exhibit certain characteristics [1-4] which are considered below.

(1) *Bit sequence independence:* The line encoder must adequately encode any source message sequence.

(2) *Transmission of adequate timing information:* The encoded bit stream must contain enough level transitions to allow for proper operation of receiver timing extraction circuitry. Transition density (average number of transitions per bit) and the maximum number of consecutive like bits are indicative of the amount of timing information within a bit sequence.

(3) *Small low frequency content:* Ac coupled optical receivers are easier to design and are capable of better performance than dc coupled receivers [18-19]. To allow for ac coupling, the received signal must contain a constant dc component. Sequences which contain an equal number of marks and spaces are called balanced sequences [20] and have a constant dc level. These balanced sequences are characterized by bounds on their consecutive like bits and accumulated digital sum variation. The tighter the bounds, the lower the short term dc fluctuation. Transmitted sequences which are unbalanced exhibit a dc fluctuation which causes a received bit error rate penalty.

(4) *High Efficiency:* Since the implementation of terminal equipment becomes more difficult as the rate of operation increases, the transmission rate must be kept as low as possible. Code efficiency is a measure of the amount of information transmitted with respect to the number of symbols used to transmit this information and is given by:

$$\text{code efficiency} = \frac{\text{\# of information symbols}}{\text{\# of transmitted symbols}} . \tag{2.1}$$

(5) *Low error multiplication:* The occurrence of a bit error during transmission should not result in many errors in the decoded bit stream. This multiplication of errors during decoding is also called error extension.

(6) *Low systematic jitter:* Jitter, a clock recovery timing noise impairment inherent to digital transmission systems, is pattern dependant [21]. The sequence of transmitted bits must ensure that jitter is kept low.

Incorporation of the following features increases the practicality of a line code.

(7) *Provision for error detection/correction:* Due to the desire for in-service monitoring of transmission system performance and automatic protection switching upon degradation of service, provision should be made to detect and possibly correct transmission errors. Integration of error detection or correction into the line code would increase system efficiency.

(8) *Provision for framing:* Although orientation of the payload within a symbol sequence can be determined by methods independent of the line code, an increase in efficiency results when the payload position is extracted by the line decoder.

(9) *Provision for ancillary channels:* Low rate communication channels additional to payload transmission are required for in-service monitoring of intermediate repeater site equipment, service signalling between terminals, and voice channels for maintenance staff. Provision for these channels within the redundancy of the line code offers a further increase in system efficiency.

(10) *Low cost:* Line code requirements should be satisfied with minimum circuit complexity, allowing benefits in terms of cost and complexity of transmission equipment to be realized.

2.3 Line Codes for Fiber Systems

Many line coding techniques which attempt to meet the requirements and offer the enhanced features described in the previous section have been proposed [1-17]. These codes can be divided into those which are unbalanced, or do not guarantee long term equality between the number of marks and spaces transmitted, and those which are balanced.

2.3.1 Unbalanced Codes

Unbalanced codes can be subdivided into codes which augment the source bit stream and those which rely on statistical randomization rather than augmentation. A number of codes including mB1C and DmB1M codes [5] fall into the first category, while the second division is commonly referred to as scrambling [6-7].

mB1C and DmB1M codes:

To form the encoded bit stream, mB1C and DmB1M codes insert a redundant bit following every m information bits. In mB1C coding, this bit has a value complementary to the m^{th} information bit. In DmB1M codes, augmentation with a mark bit is followed by a finite sum operation which translates a mark to a change of encoded stream level and a space to the absence of a level change. Both codes enforce a maximum consecutive bit run length of $(m+1)$ and ensure at least one transition every $(m+1)$ bits. DmB1M codes multiply one transmission error to two during decoding, while mB1C codes exhibit no error multiplication. Code rule violations allow both codes to estimate the decoded error rate.

Even with their unbalanced form and corresponding penalty in received bit error rate, the simple encoding and decoding procedures of these codes have resulted in widespread use in high bit rate transmission systems.

Scrambling:

Scrambling denotes a probabilistic approach to line coding which ensures no limit to either the maximum consecutive like bits or digital sum variation which can occur in the transmitted bit stream. This bit stream randomization without augmentation is performed through either reset or self-synchronizing scrambling.

Reset scrambling involves the modulo-2 addition of the source bit stream with a known sequence. This known sequence is usually pseudo-random, producing an encoded stream which is as random as possible. Pseudo-random sequences can be easily generated through continuous cycling of a shift register implementing polynomial division. The divisor is usually chosen to be a primitive irreducible polynomial since these polynomials generate the longest pseudo-random sequences possible with shift register implementation. A primitive irreducible polynomial of degree d will generate a sequence which repeats only after (2^d-1) symbols. A partial listing of these polynomials is given in Table 2.1. A full listing of these polynomials can be found in Peterson and Weldon [22]. A circuit which generates a 127 bit pseudo-random sequence is given in Figure 2.2.

The original source stream is recovered at the decoder through modulo-2 addition of the received bit stream with the same pseudo-random sequence. Although the correct sequence is known, this modulo-2 addition will correctly cancel the encoding addition only if the sequence is added at the correct time. The required synchronization can be achieved through resetting the shift register to a pre-determined non-zero value at a time derived from the framing signal. Implementation of a seven stage reset scrambler is depicted in Figure 2.3.

Reset scrambling exhibits no error extension but also makes no provision for error monitoring.

Degree	Polynomial	Degree	Polynomial
2	x^2+x+1	9	x^9+x^4+1
3	x^3+x+1	10	$x^{10}+x^3+1$
4	x^4+x+1	11	$x^{11}+x^2+1$
5	x^5+x^2+1	12	$x^{12}+x^6+x^4+x+1$
6	x^6+x+1	13	$x^{13}+x^4+x^3+x+1$
7	x^7+x^3+1	14	$x^{14}+x^{10}+x^6+x+1$
8	$x^8+x^4+x^3+x^2+1$	15	$x^{15}+x+1$

Table 2.1: Primitive Irreducible Polynomials

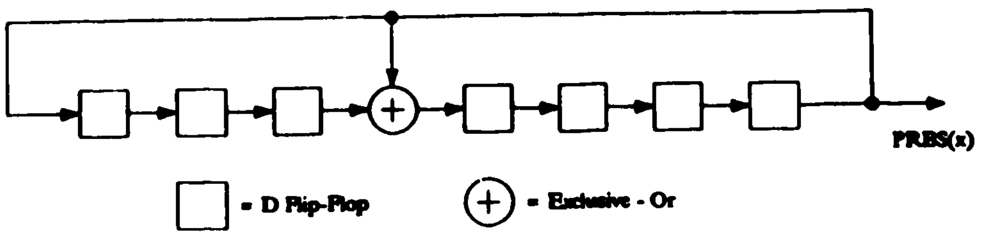


Figure 2.2: Shift Register Generation of a Pseudo-random Bit Sequence.
Sequence length = 127. $d(x) = x^7 + x^3 + 1$.

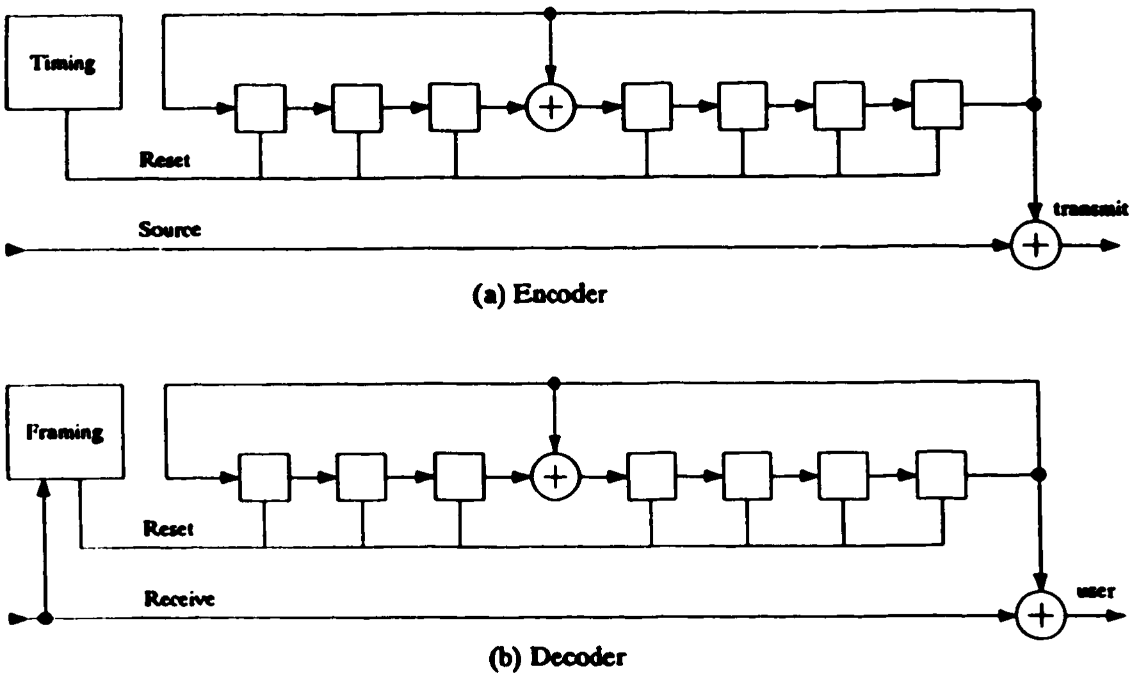
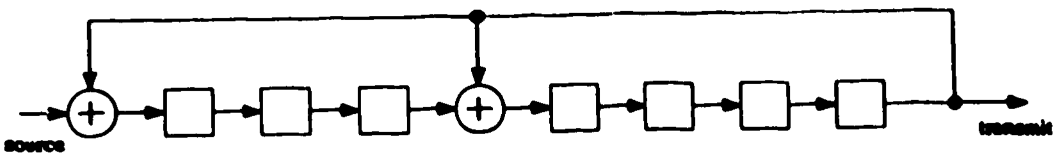
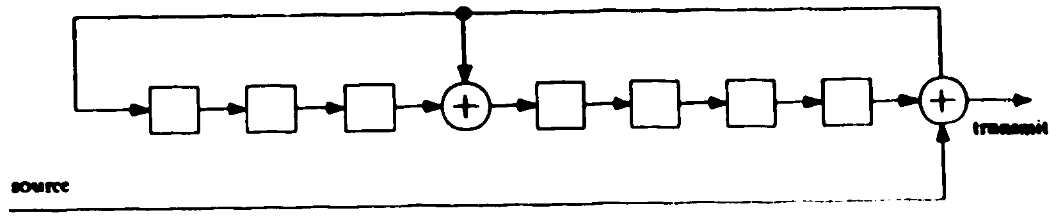


Figure 2.3: Reset Scrambling with 7 stage PRBS generation



(a) With 7 bit period delay



(b) Without delay

Figure 2.4: Shift Register Polynomial Division, with and without delay.
$d(x) = x^7 + x^3 + 1$

The name *self-synchronizing scrambling* is misleading. Rather than being self-synchronizing, the encoding and decoding procedures of this line code operate without synchronization. Instead of describing this technique with the usual time domain expressions [6], encoding can be viewed simply as the continuous division of the source bit stream by a polynomial. To enhance stream randomness, the divisor chosen is usually a primitive irreducible polynomial, enabling a single mark in the source stream to produce an encoded signal which is pseudo-random in nature. Shift register implementations for polynomial division, with and without processing delay, are given in Figure 2.4.

In the absence of transmission errors, continuous multiplication by the same polynomial at the decoder accurately restores the original source bit stream. An example of a shift register which performs this multiplication is given in Figure 2.5. Self-synchronizing scrambling is usually implemented with circuits similar to those in Figures 2.4(b) and 2.5 to encode and decode without delay. Since both the division and multiplication operations are continuous, there is no need for synchronization.

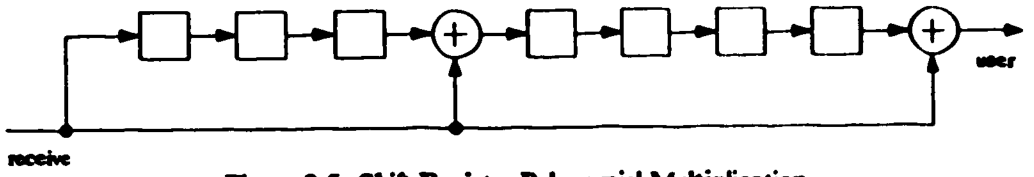


Figure 2.5: Shift Register Polynomial Multiplication.
$d(x) = x^7 + x^3 + 1$

The feed-forward nature of decoder polynomial multiplication causes self-synchronizing scrambling to exhibit error extension. Each transmission error will cause the incorrect decoding of a r of bits equal to the weight of the scrambling polynomial, where the weight of a polynomial is to the number of its non-zero coefficients. However, if the transmission errors are spaced such that the extended errors occupy the same bit position, the extended errors cancel through modulo-2 addition. The upper bound for error extension becomes:

$$\# \text{ of decoded errors} \leq (\# \text{ of transmission errors}) * (\text{weight of scrambling polynomial}). \quad (2.2)$$

Self-synchronizing scrambling makes no provision for error monitoring.

Although the self-synchronizing encoding and decoding procedures can be reversed, this arrangement is not practical. Encoding through polynomial multiplication will introduce randomness into the source stream, and subsequent polynomial division will accurately restore the source bit stream in the absence of transmission errors. However, the feedback nature of polynomial division can cause the decoder to generate an unlimited number of errors in the decoded bit sequence in response to even a single transmission error.

Because reset and self-synchronizing scrambling can be implemented with very simple logic circuits which introduce no bit stream delay, scrambling has enjoyed widespread use in high bit rate systems. However, scrambling alone is generally not considered to be adequate and is usually implemented in conjunction with other line coding techniques.

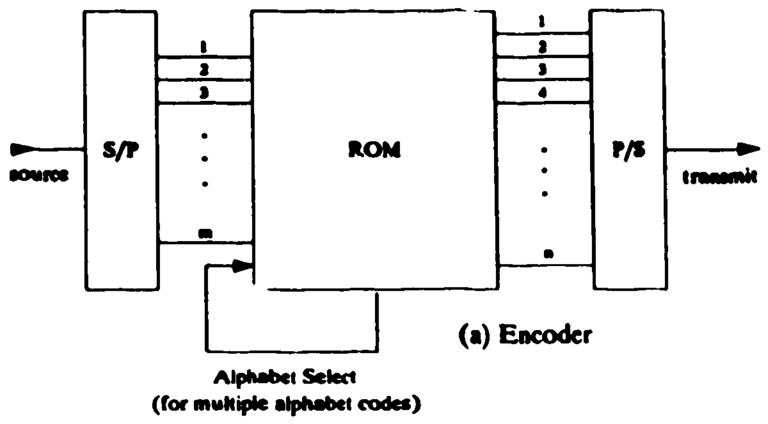
2.3.2 Balanced Codes

To force the encoded bit stream to be balanced while at the same time retaining source stream bit sequence independence, the source stream must be augmented. The method in which this augmentation is implemented distinguishes the family of alphabetic codes from other forms of balanced coding.

Alphabetic Codes:

A great deal of effort has been directed towards the development of codes which map m information bits into n transmission bits (n>m) where the mapping is performed through look-up tables [2,8-12]. Such codes are designed by pre-selecting a set or sets of n-bit codewords, with set selection usually based on low word disparity. Each set of words is called an "alphabet", the codes "alphabetic" [23]. Each m-bit source word is assigned an n-bit representation from each alphabet. The alphabet from which a codeword is selected is determined by the "state" of the encoder. These codes are commonly referred to as mBnB codes, where B denotes a binary signal.

A block diagram of an encoder and decoder which implement the look-up tables with read only memory (ROM) is given in Figure 2.6. This universal form of alphabetic code implementation imposes limitations on the rate of operation and code efficiency. ROM I/O is typically slower than



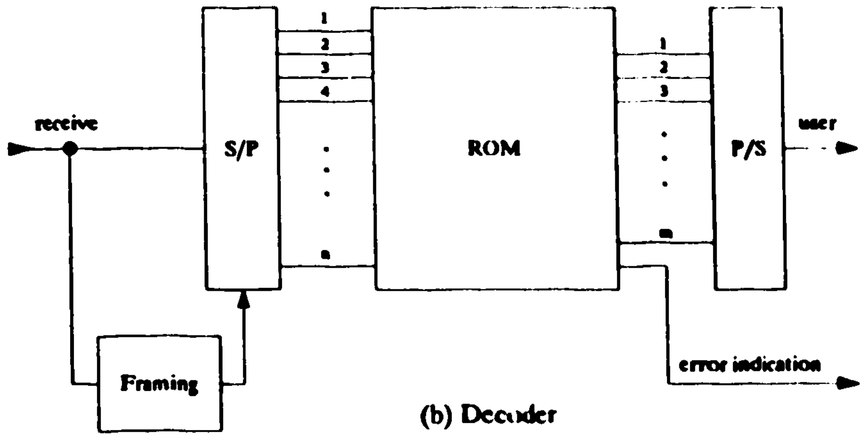


Figure 2.6: Alphabetic Code Implementation

sequential or combinational logic operations, and the physical limitation of ROM capacity enforces a limit to wordlength and code efficiency. For these reasons, alphabetic codes have not gained popularity in high bit rate systems. 5B6B, 7B8B and 10B12B codes are in use in lower rate systems today. As VLSI technology advances, these limitations will decrease and alphabetic codes might be used in high rate systems.

Alphabetic codes exhibit error extension. A single error can cause the selection of a wrong word at the decoder, resulting in as many as m decoded errors. But these codes can also indicate when an illegal word has been received, providing good error monitoring at low bit error rates.

Non-Alphabetic Codes:

Many methods have also been proposed for balanced line coding through simple encoding and decoding operations. However, the majority of these proposals have been directed towards highly inefficient codes, including codes which transform one source bit to two encoded bits [3-4,13] or impose strict limitations on encoded word length [14-15]. [illegible] inefficiency of these codes precludes their use in systems of high source bit rate.

A simple balanced line coding technique which can be highly efficient was proposed by Carter in 1965 [16]. This technique ensures that the transmitted bit sequence will be balanced by transmitting either the source word or its complement, whichever reduces the disparity of the transmitted bit stream. An extra bit which is concatenated to each source word indicates whether or not this word has been inverted, allowing for recovery of the original signal at the decoder.

But since the pioneering work of Carter, relatively little effort has been directed towards the development of highly efficient, easily implemented, balanced line codes. A notable exception is Krzymien's family of Partially Flipped codes which, while based on the same principle, perform slightly better than Carter's codes [17]. The next chapter introduces a new, balanced, non-alphabetic line coding technique which can be implemented with high speed combinational and sequential logic circuits. Its insurance of low error extension and high encoded stream transition density makes this technique attractive for high bit rate fiber system implementation.

CHAPTER 3
THE CONCEPT OF GUIDED SCRAMBLING

This chapter outlines the basis for Guided Scrambling. A new interpretation of self-synchronizing scrambling processes reveals a property which can be exploited to enhance the transmitted bit stream characteristics, and in doing so introduces the concept of Guided Scrambling.

3.1 Mathematics of Self-synchronizing Scrambling

The concept of Guided Scrambling is revealed through examination of self-synchronizing scrambling and unscrambling procedures. Previous descriptions of these procedures have modelled the scrambling and unscrambling circuits as binary sequence filters and have limited their mathematical description to equations representative of time-domain circuit operation [6-7, 25]. A simpler description can be developed when it is realized that the circuits used in scrambling are also used in error-correction techniques to implement polynomial division and multiplication [26]. This allows self-synchronizing scrambling to be interpreted as the continuous division of the source bit sequence by a polynomial and unscrambling as multiplication of the transmitted quotient by the same polynomial.

A mathematical description becomes straightforward when the bit streams are represented by polynomials and the rules of modulo-2 arithmetic given in Table 3.1 are applied. In accordance with notation established in Section 1.2, let:

$s(x)$	=	the source bit stream,
$d(x)$	=	the scrambling polynomial, of degree d,
$q(x)$	=	the quotient of encoding division,
$tx(x)$	=	the transmitted signal,
$e(x)$	=	the error pattern introduced during transmission,
$rx(x)$	=	the received bit stream,
$u(x)$	=	the decoded bit stream.

a	b	a + b	ab	a / b
0	0	0	0	0, remainder 0
0	1	1	0	0, remainder 1
1	0	1	0	undefined
1	1	0	1	1, remainder 0

Table 3.1: Modulo-2 Arithmetic Operations

Self-synchronizing scrambling division forms the transmitted quotient stream:

$$tx(x) = q(x) = Q_{d(x)}[s(x)\, x^d] \, , \tag{3.1}$$

where Q denotes the formation of a quotient through division of its argument by its subscripted polynomial. Premultiplication of the source sequence by x^d yields delayless encoding.

When this sequence becomes corrupted during transmission, the received sequence can be written:

$$rx(x) = tx(x) + e(x) \, . \tag{3.2}$$

Delayless multiplication of this received sequence by the scrambling polynomial and disregard of the d least significant bits of the product yields the decoded bit sequence:

$$\begin{aligned} u(x) &= Q_{x^d}[rx(x)\, d(x)] \\ &= Q_{x^d}[(tx(x) + e(x))\, d(x)] \\ &= s(x) + Q_{x^d}[e(x)\, d(x)] \, . \end{aligned} \tag{3.3}$$

While Equation (3.3) depicts the accurate recovery of the original source stream in the absence of errors, it also demonstrates that transmission errors are multiplied by the scrambling polynomial during decoding.

3.2 Basis of Guided Scrambling

Close examination of the self-synchronizing scrambling encoding process reveals an interesting property, namely that the value of each source bit can affect the values of many quotient bits. Consider the following example which uses the shorthand polynomial representation introduced in Section 1.2.1. When:

$$\begin{aligned} s_0(x) &= 01001010001001010100101 \\ d(x) &= 100101 \, , \end{aligned}$$

the transmitted quotient would be:

$$\begin{aligned} q_0(x) &= Q_{d(x)}[s_0(x)\, x^d] \\ &= 01000000001000000100000 \, . \end{aligned}$$

This unbalanced sequence with few transitions does not satisfy line code criteria. However, if the first bit of $s_0(x)$ had been a mark instead of a space, as:

$$s_1(x) = 11001010001001010100101 \, ,$$

division by the same d(x) would result in the quotient:

$$q_1(x) = 11010110010111000010111 \, .$$

This sequence is much better suited for transmission. Figure 3.1 shows long division depicting the

```
            010000000001000000100000   = q_0(x)
100101 |0100101000100101010010100000   = s_0(x) x^5
   "     100101
         ------
d(x)     000000000100101
                  100101
                  ------
                  0000000100101
                         100101
                         ------
(a) Derivation of q_0(x) 00000000000   = remainder
```

```
             11010110010111000010111   = q_1(x)
100101 |1100101000100101010010100000   = s_1(x) x^5
   "    100101
        ------
d(x)     101111
         100101
         ------
           101000
           100101
           ------
             110101
             100101
             ------
              100000
              100101
              ------
                 101010
                 100101
                 ------
                   111110
                   100101
                   ------
                    110111
                    100101
                    ------
                     100100
                     100101
                     ------
                          101010
                          100101
                          ------
                            111100
                            100101
                            ------
                             110010
                             100101
                             ------
                              101110
                              100101
                              ------
(b) Derivation of q_1(x)       01011   = remainder
```

Figure 3.1: Polynomial Division

generation of both $q_0(x)$ and $q_1(x)$.

An algebraic description of modulo-2 division provides an explanation for the drastically different quotient streams. Let:

$$s_0(x) = c_{j-1}x^{j-1} + c_{j-2}x^{j-2} + \dots + c_kx^k + \dots + c_1x + c_0$$

$$q_0(x) = Q_{d(x)}[s_0(x)\, x^d] \;.$$

Let $s_1(x)$ be equal to $s_0(x)$, except for complementation of the coefficient c_k:

$$\begin{aligned} s_1(x) &= c_{j-1}x^{j-1} + c_{j-2}x^{j-2} + \ldots + \bar{c}_k x^k + \ldots + c_1 x + c_0 \\ &= s_0(x) + x^k . \end{aligned}$$

Then:

$$\begin{aligned} q_1(x) &= Q_{d(x)}[s_1(x)\, x^d] \\ &= Q_{d(x)}[(s_0(x) + x^k)\, x^d] \\ &= q_0(x) + Q_{d(x)}[x^{k+d}] . \end{aligned} \tag{3.4}$$

The term $Q_{d(x)}[x^{k+d}]$ may contain many non-zero coefficients. The modulo-2 addition of this sequence with $q_0(x)$ will result in a quotient stream $q_1(x)$ with bit values complementary to those of $q_0(x)$ wherever a non-zero coefficient of $Q_{d(x)}[x^{k+d}]$ occurs. To continue with the last example, straightforward arithmetic yields:

$$\begin{aligned} s_1(x) &= s_0(x) + x^{22} \\ Q_{d(x)}[x^{22+5}] &= 10010110011111000110111 , \end{aligned}$$

and:

$$\begin{array}{rl} q_0(x) & 01000000001000000100000 \\ + \underline{Q_{d(x)}[x^{27}]} & \underline{10010110011111000110111} \\ q_1(x) & 11010110010111000010111 . \end{array}$$

Note that the vastly different $q_0(x)$ and $q_1(x)$ bit sequences are restored to $s_0(x)$ and $s_1(x)$ with the same multiplication circuit even though $s_0(x)$ and $s_1(x)$ differ in only their most significant bit. A shift and add representation of this multiplication is given in Figure 3.2.

```
  q0(x)      01000000001000000100000
• d(x)                        100101
------------------------------------
                          10010100000
                   10010100
         01001010000
------------------------------------
  s0(x)  01001010001001010100101■■■■■

(a) Restoration of s0(x)

■ = discarded through simultaneous
    division by x^5
```

```
  q1(x)     11010110010111000010111
• d(x)                       100101
-----------------------------------
                             100101
                            100101
                           100101
                         100101
                     100101
                    100101
                   100101
                 100101
              100101
             100101
            100101
           100101
          100101
-----------------------------------
  s1(x)  11001010001001010100101■■■■■

(b) Restoration of s1(x)
```

Figure 3.2: Polynomial Multiplication

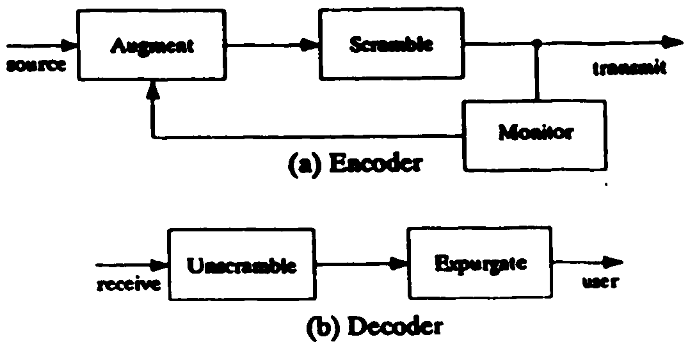


Figure 3.3: Conceptual diagram of Guided Scrambling.

With the observation that the value of a single dividend bit can so radically alter the appearance of its quotient, that the dividend can be recovered from the quotient simply through multiplication, and that both polynomial division and multiplication can be implemented with high-speed logic circuits, the following questions were asked:

Can an unconstrained high rate binary source stream be augmented prior to self-synchronizing scrambling such that the transmitted bit stream contains the characteristics of a good line code? Will simple removal of the augmenting bits from the unscrambled sequence restore the original bit stream?

The concepts of these questions are captured in Figure 3.3.

Through analysis and computer simulation it has been determined that appropriate augmentation of the source stream will guide the scrambling process to produce a balanced quotient bit stream with a high transition density, and that proper updating of the shift registers will allow for recovery of the source stream in the suggested manner. In consideration of its similarity to self-synchronizing scrambling and its ability to guide this process to ensure satisfaction of line code requirements, this new line coding technique has been called Guided Scrambling. The next chapter discusses configuration alternatives for GS coding.

CHAPTER 4
GUIDED SCRAMBLING CONFIGURATIONS

Guided Scrambling encoding can be efficiently implemented with a structure similar to that depicted in Figure 4.1. In this implementation, GS encoding proceeds in the following manner:

(1) Augment the incoming source stream with identically placed, dissimilar bits to generate a number of different augmented bit streams.

(2) Scramble each augmented sequence through simultaneous multiplication by x^d and division by d(x) to create a set of quotient sequences which represent the same source word.

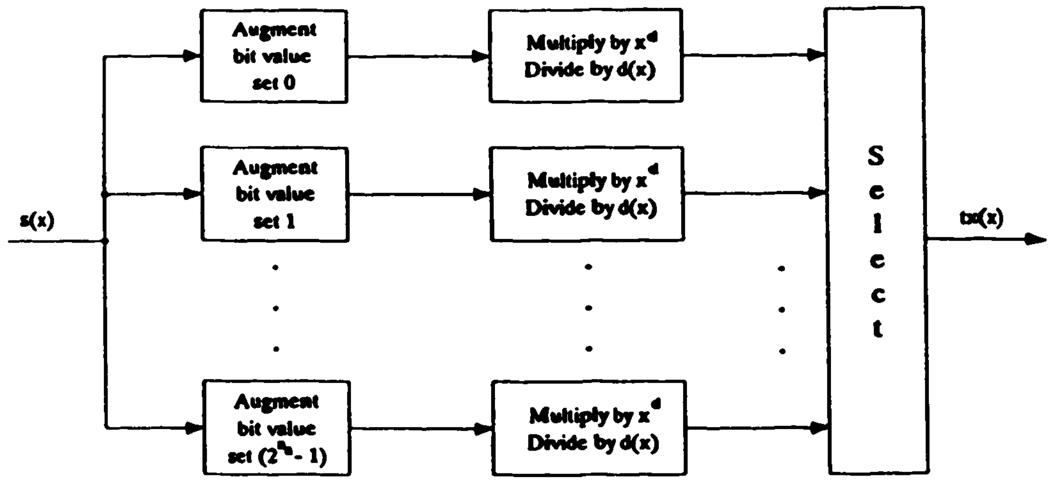
(3) Select the quotient sequence which best satisfies line code requirements.

(4) Transmit the selected quotient sequence.

Although premultiplication by x^d allows for quotient generation without delay, a quotient selection mechanism which requires full knowledge of the quotient alternatives introduces a one word encoding delay while conceptually providing the feedback which guides the scrambling process. To decode the transmitted bit stream without delay:

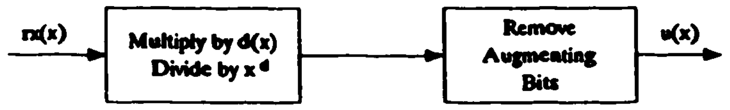
(1) Multiply the received sequence by d(x) and discard the d least significant bits of the product.

(2) Remove the augmenting bits from the product bit stream.



s(x)
Augment bit value set 0
Augment bit value set 1
Augment bit value set (2^a - 1)
Multiply by x^d Divide by d(x)
Select
tx(x)

(a) Encoder



(b) Decoder

Figure 4.1: Efficient Implementation of Guided Scrambling

In outlining the general Guided Scrambling structure, this algorithm poses several specific questions concerning Guided Scrambling implementation which must be individually addressed. A number of GS configurations result from considering the following questions:

(1) In what pattern should the augmenting bits be inserted into the source bit stream?

(2) What scrambling polynomials ensure that the transmitted stream will contain the characteristics of a good line code?

(3) On what merits should quotients be selected?

(4) Must the encoding and decoding registers be updated following quotient selection and source word recovery to provide accurate restoration of the source bit stream?

This chapter discusses these questions in detail.

4.1 Placement of Augmenting Bits

Augmenting bits can be placed in fixed or varying positions within the source bit stream. Although insertion of bits into the source bit stream without a regular distribution might result in an encoded stream with better line code characteristics, variable placement also has several disadvantages including:

(1) an increase in encoder circuit complexity since a decision must be made regarding both the value and placement of augmenting bits.

(2) a reduction in line code efficiency due to the necessity of transmitting information regarding the placement of augmenting bits.

(3) loss of frame or catastrophic error extension resulting from transmission errors which alter augmenting bit placement information and cause incorrect bit removal.

Given these disadvantages, only regular placement of augmenting bits has been explored for Guided Scrambling.

The insertion of augmenting bits in fixed, regularly spaced bit positions allows for analysis, implementation, and description of Guided Scrambling as a constant length block code. In particular, when these codes view the incoming source stream as a series of words m bits in length and translate each source word into a codeword of length n, the notation GSmBnB appropriately describes the coding technique.

As shown in Figure 4.1, the proposed encoding structure augments each m-bit source word with n_a augmenting bits to give an encoded wordlength of:

$$n = m + n_a \,. \tag{4.1}$$

The introduction of all combinations of augmenting bit values generates 2^{n_a} augmented sequences from each source word. Division of these augmented sequences by the scrambling polynomial produces 2^{n_a} quotients which collectively form a quotient selection set. From this set a word must be selected for transmission. In accordance with the definitions of Section 1.2.1, let $a_i(x)$ denote the augmented sequence where the decimal value of the augmenting bits is i, and let $q_i(x)$ denote the quotient it generates.

Since augmenting bit values influence the statistics of the quotient sequences to the greatest extent when they are inserted prior to the m-bit source word, Guided Scrambling analysis has involved placement of the augmenting bits only in the most significant bit positions.

4.2 Potential Scrambling Polynomials

To ensure that the transmitted bit stream will contain the characteristics of a good line code, the scrambling polynomial must be chosen such that there will exist in every quotient selection set a quotient which exhibits these characteristics. Derivation of scrambling polynomials appears difficult because there is limited information on the statistics of quotients which appear in the selection sets. Indeed, as found in the first section of Appendix 1, when no restrictions are placed on the form of the source bit stream and all combinations of augmenting bit values are introduced into each source word, quotients in the selection set can take on any form.

But derivation of potential scrambling polynomials is made possible by investigating the relationships between quotients in the selection set which are independent of quotient statistics. Line code requirements translate to restrictions on their form. This section considers first the relationships among quotients in the selection set and then derives scrambling polynomials which provide good line code characteristics when $n_a = 1$. The principles of this derivation are then extended to consideration of scrambling polynomials for multiple bit augmentation.

4.2.1 Quotient Set Relationships

As shown in Section 3.3, quotients can be related through modulo-2 addition with sequences which depend on the positions of differing dividend bits. In Guided Scrambling, these relationship sequences remain constant since dividends differ only in the augmenting bit positions. In the case of a single augmenting bit of degree (n-1), the quotients $q_0(x)$ and $q_1(x)$ can be related by rewriting Equation (3.4) as:

$$\begin{aligned} q_1(x) &= q_0(x) + Q_{d(x)}[x^{(n-1)+d}] \\ &= q_0(x) + r(x) \ , \end{aligned} \tag{4.2}$$

where:

$$r(x) = Q_{d(x)}[x^{(n-1)+d}] \ . \tag{4.3}$$

Quotients which result from the division of sequences augmented with multiple bits can be similarly related. Let their relationships be denoted by the sequences $r_i(x)$, where $r_i(x)$ is the n-bit sequence which relates $q_i(x)$ and $q_0(x)$. With this notation, $r_0(x)$ is the all zero sequence. To

Augmenting bits (degrees 7 and 6)	Polynomial Expression	Bit Stream Representation
0 0	$r_0(x) = 0$	$r_0(x) = 00000000$
0 1	$r_1(x) = Q_{d(x)}[x^8]$	$r_1(x) = 01010101$
1 0	$r_2(x) = Q_{d(x)}[x^9]$	$r_2(x) = 10101010$
1 1	$r_3(x) = Q_{d(x)}[x^9+x^8]$	$r_3(x) = 11111111$

Table 4.1: $r_i(x)$ for $n = 8$, $n_a = 2$, $d(x) = x^2 + 1$.

demonstrate, Table 4.1 depicts the $r_i(x)$ sequences when $n=8$, $n_a=2$, $d(x) = x^2+1$.

It is clear from Equation (4.3) and Table 4.1 that the sequences which relate quotients in the selection set depend only on the placement of augmenting bits and the scrambling polynomial. By restricting augmenting bit placement to the most significant bit positions and deriving limitations on the form of the relationship sequences, potential Guided Scrambling polynomials can be established.

4.2.2 Scrambling polynomial for high encoded stream transition density, $n_a = 1$

The quotient selection set generated when augmentation is with a single bit per word contains two quotients. As derived in Appendix 1, when the sequence r(x) which relates these two quotients contains $t_{r,int}$ transitions, one of the quotients will contain t_{int} transitions where:

$$t_{int} \geq \begin{cases} \dfrac{t_{r,int} + 1}{2}, & t_{r,int} = \text{odd} \\[2ex] \dfrac{t_{r,int}}{2}, & t_{r,int} = \text{even}, \end{cases} \qquad (4.4)$$

and t_{int} refers to the number of transitions within a bit sequence exclusive of the transition which might occur prior to the most significant bit. It is clear then that the amount of timing information transmitted can be maximized by maximizing the number of transitions within r(x). An alternating n-bit sequence contains the maximum of (n-1) internal transitions. When r(x) is this alternating nce, a quotient can always be selected with:

$$t_{int} \geq \begin{cases} \dfrac{n}{2}, & n = \text{even} \\[2ex] \dfrac{n-1}{2}, & n = \text{odd}. \end{cases} \qquad (4.5)$$

Figure 4.2, the required alternating sequence is generated from the division of a single

```
        101010101...     = alternating sequence
101 | 10000000000...
      101                Dividend = one non-zero
       100                          coefficient.
       101
        100              d(x) = x²+ 1.
        101
         100
         101
          100...
```

Figure 4.2: Generation of the alternating sequence.

non-zero coefficient polynomial by $d(x) = x^2 + 1$. This scrambling polynomial allows for transmission of the most timing information when $n_s = 1$.

4.2.3 Scrambling polynomials for balanced transmission, $n_s = 1$

By definition, balanced transmission is ensured when the encoded stream running digital sum is bounded. Bounds on the RDS of the transmitted sequence imply that bounds also exist on its word-end running digital sum. As shown in Appendix 1, enforcement of encoded stream WRDS bounds requires that n_{nc}, the number of $q_0(x)$ bits not complemented through modulo-2 addition with $r(x)$ to form $q_1(x)$ must be:

$$n_{nc} = \begin{cases} 0, & n = \text{odd} \ , \\ 0 \text{ or } 1, & n = \text{even} \ . \end{cases} \tag{4.6}$$

Given these values for n_{nc}, sequences which relate quotients in the selection set and allow for balanced transmission are:

$$r(x) = \quad 1111\ldots11 \ , \qquad n = \text{odd} \ ,$$
$$r(x) = \begin{cases} 1111\ldots11 \\ 0111\ldots11 \\ 1011\ldots11 \\ 1101\ldots11 \ , \\ \vdots \\ 1111\ldots01 \\ 1111\ldots10 \end{cases} \qquad n = \text{even} \ . \tag{4.7}$$

Not all of these sequences can be generated through the division of a polynomial with a single non-zero coefficient by d(x). The sequences which can be generated are given in Table 4.2 along with their associated minimum weight scrambling polynomial. Note that a balanced encoded bit stream can be generated for both even and odd length quotient sequences.

The family of scrambling polynomials associated with r(x) = 1011...11 are not as complicated as their polynomial expressions in Table 4.2 might indicate. When written in bit sequence form in Table 4.3, these polynomials demonstrate a consistent pattern: the most significant non-zero

r (x)	Associated Scrambling Polynomial d(x)	
1111 ... 11	$x + 1$	
1011 ... 11	$x^{n-1} + x^{n-3} + x^{n-4} + x^{n-6} + x^{n-7} + \ldots + 1$	n mod 6 = 0
	$x^{n-2} + x^{n-4} + x^{n-5} + x^{n-7} + x^{n-8} + \ldots + x + 1$	n mod 6 = 2
	$x^{n-1} + x^{n-3} + x^{n-4} + x^{n-6} + x^{n-7} + \ldots + x + 1$	n mod 6 = 4
1111 ... 01	$x^{n-2} + x^{n-3} + 1$	n = even
1111 ... 10	$x^{n-1} + x^{n-2} + 1$	n = even

Table 4.2: Minimum Weight Scrambling Polynomials for Balanced Guided Scrambling, $n_a = 1$.

n	d(x) (msb ... lsb)
4	1011
6	101101
8	1011011
10	1011011011
12	101101101101
14	1011011011011

Table 4.3: Bit Sequence Representation of Scrambling Polynomials associated with r(x) = 1011 ... 11.

coefficient is followed by a repetitive pattern of one zero and two non-zero coefficients. To form an r(x) of the required length, this repetitive sequence must be truncated at a point which depends on factorization of quotient length by six. The polynomial expressions in Table 4.2 reflect this dependence. For future convenience, this family of scrambling polynomials will be denoted $D_{mod6}(x)$, where capitalization denotes reference to a group of polynomials and the subscript marks the dependance of polynomial form on encoded wordlength factorization by six.

4.2.3 Potential scrambling polynomials for multiple bit augmentation

It is clear from the r(x) sequences composed entirely of ones in (4.7) that a sufficient condition for the generation of a balanced encoded bit stream is the presence of a complementary pair of quotients in the quotient selection set. That this condition is sufficient is not surprising since it is the foundation for Carter's codes [16].

If the quotient selection set consists of a number of pairs of complementary words, it is clear that RDS bounds can be enforced with the selection of a number of different alternatives under any circumstance. Selection from this subset of words which enforces RDS bounds can be based on another criterion such as the number of transitions, allowing for an increase in the amount of timing information transmitted while maintaining balanced transmission. The use of multiple augmenting bits and the appropriate d(x) generates multiple sets of complementary quotients.

Noting that modulo-2 addition has the property:

$$\begin{aligned} \text{if} \quad & q_0(x) + r_i(x) = q_i(x) \\ & q_0(x) + r_j(x) = q_j(x) \\ \text{and} \quad & r_i(x) = \overline{r_j(x)} \qquad (4.8) \\ \text{then} \quad & q_i(x) = \overline{q_j(x)} \, , \end{aligned}$$

the presence of pairs of complementary quotient sequences in the selection set is ensured if it can be shown that d(x) generates complementary relationship sequences. It is straightforward to show that $d(x) = x^{n_a}+1$ generates the maximum 2^{n_a-1} pairs of complementary quotients in a 2^{n_a} member quotient selection set. As shown in Tables 4.4 through 4.6, this divisor generates $r_i(x)$ sequences which are a repetition of the corresponding augmenting bit pattern. When all 2^{n_a-1} complementary augmenting bit patterns are used to generate 2^{n_a} quotients, the selection set will then consist of 2^{n_a-1} pairs of complementary quotients. Accordingly, $d(x) = x^{n_a}+1$ is proposed as a scrambling polynomial for multiple bit augmentation.

Augmenting bits	Relationship sequence	Complementary $r_i(x)$
00	$r_0(x)$ = 0000000000	$r_3(x)$
01	$r_1(x)$ = 0101010101	$r_2(x)$
10	$r_2(x)$ = 1010101010	$r_1(x)$
11	$r_3(x)$ = 1111111111	$r_0(x)$

Table 4.4: Quotient relationships.
$n = 10, n_a = 2, d(x) = x^2 + 1.$

Augmenting bits	Relationship sequence	Complementary $r_i(x)$
000	$r_0(x)$ = 0000000000	$r_7(x)$
001	$r_1(x)$ = 0010010010	$r_6(x)$
010	$r_2(x)$ = 0100100100	$r_5(x)$
011	$r_3(x)$ = 0110110110	$r_4(x)$
100	$r_4(x)$ = 1001001001	$r_3(x)$
101	$r_5(x)$ = 1011011011	$r_2(x)$
110	$r_6(x)$ = 1101101101	$r_1(x)$
111	$r_7(x)$ = 1111111111	$r_0(x)$

Table 4.5: Quotient relationships.
$n = 10, n_a = 3, d(x) = x^3 + 1.$

Augmenting bits	Relationship sequence	Complementary $r_i(x)$
0000	$r_0(x)$ = 0000000000	$r_{15}(x)$
0001	$r_1(x)$ = 0001000100	$r_{14}(x)$
0010	$r_2(x)$ = 0010001000	$r_{13}(x)$
0011	$r_3(x)$ = 0011001100	$r_{12}(x)$
0100	$r_4(x)$ = 0100010001	$r_{11}(x)$
0101	$r_5(x)$ = 0101010101	$r_{10}(x)$
0110	$r_6(x)$ = 0110110110	$r_9(x)$
0111	$r_7(x)$ = 0111011101	$r_8(x)$
1000	$r_8(x)$ = 1000100010	$r_7(x)$
1001	$r_9(x)$ = 1001001001	$r_6(x)$
1010	$r_{10}(x)$ = 1010101010	$r_5(x)$
1011	$r_{11}(x)$ = 1011101110	$r_4(x)$
1100	$r_{12}(x)$ = 1100110011	$r_3(x)$
1101	$r_{13}(x)$ = 1101110111	$r_2(x)$
1110	$r_{14}(x)$ = 1110111011	$r_1(x)$
1111	$r_{15}(x)$ = 1111111111	$r_0(x)$

Table 4.6: Quotient relationships.
$n = 10, n_a = 4, d(x) = x^4 + 1.$

4.3 Quotient Selection Criteria

In GS encoding, each transmitted quotient is selected from a number of alternatives. A basis for this selection must be established. Noting that encoding must be bit sequence independent and should be capable of high bit rate operation, other line code requirements given in Section 2.2 include the generation of a transmission sequence with adequate timing information, a dc balance, high efficiency and low systematic jitter, and a decoder which introduces low error multiplication.

Since efficiency is a function of the introduced redundancy and error multiplication is a function of the weight of the scrambling polynomial, neither can be used for encoder quotient selection. A word by word selection device also cannot easily make a decision regarding the systematic jitter of a transmitted sequence. The two remaining selection criteria include consideration of the transmission of adequate timing information and the enforcement of RDS bounds. Selection mechanisms based on these criteria are discussed below.

Select the quotient with the greatest number of transitions:

An encoder quotient selection mechanism could select the sequence which contains the greatest number of transitions. This technique ensures that the transmitted stream will contain the greatest amount of timing information given the other code parameters. Unfortunately, this mechanism will not guarantee a balanced encoded bit stream. Consider selection between the alternatives given below. The number of transitions t assumes that the last transmitted bit was a 0.

$$q_0(x) \ : \ 0011101110 \quad t = 4 \quad WD = 2$$
$$q_1(x) \ : \ 1001000100 \quad t = 6 \quad WD = -4 \ .$$

If this set of alternatives occurs repetitively and if selection is based solely on the number of transitions, $q_1(x)$ will continually be chosen. Given the negative value of its word disparity, the encoded stream running digital sum would decrease without bound, and an unbalanced encoded bit stream would result. This selection technique should then be considered only when balance of the transmitted sequence is not of concern.

Select the quotient with minimum $|RDS|$:

In an attempt to produce a balanced encoded signal, an encoder could select the quotient which exhibits the least running digital sum disparity. Since this selection method clearly does not ensure that the word with the most number of transitions will be selected, it is expected that other decision methods would increase the amount of timing information transmitted.

Surprisingly, this selection mechanism also cannot ensure balanced transmission. Consider the following set of alternatives. Let RDS′ denote the RDS of the transmitted sequence prior to selection from this quotient set.

$q_0(x)$: 0101101011 $|RDS|_{max}$ = RDS′ + 2 WD = 2

$q_1(x)$: 1111000001 $|RDS|_{max}$ = RDS′ + 4 WD = 0 .

If RDS′ is greater than zero, selection of the word with minimum $|RDS|_{max}$ would result in the transmission of $q_0(x)$. If these alternatives or alternatives with similar characteristics occur repetitively, the positive disparity of the selected word would cause the encoded stream running digital sum to grow without bound.

Select the quotient with minimum | WRDS |:

If the scrambling polynomial is chosen such that WRDS bounds can be enforced, selection of the alternative with the minimum |WRDS| will enforce these bounds. But once again, another selection mechanism might provide for a higher encoded stream transition density. Consider selection between the following alternatives, assuming WRDS prior to word selection is 0:

$q_0(x)$: 0101101011 WRDS = 2 t_{int} = 7

$q_1(x)$: 1111000001 WRDS = 0 t_{int} = 2 .

If the word with minimum |WRDS| is selected, $q_1(x)$ would be chosen. But if it is known that a WRDS bound greater than 2 can always be enforced, a different selection mechanism could select the sequence with five more transitions and still provide balanced transmission.

Select the quotient with the greatest number of transitions, given enforcement of WRDS bounds:

If the scrambling polynomial is chosen such that WRDS bounds can be enforced, selection of the alternative with the greatest number of transitions, given that it does not violate established WRDS bounds, will produce a balanced encoded bit stream with the greatest number of level transitions in the Guided Scrambling process. Conversely, simulation results reported in the next chapter demonstrate that this mechanism increases the length and number of occurrences of long like-bit sequences in some GS configurations.

Select the quotient with a transition at a specific transition opportunity, given enforcement of WRDS bounds:

Selection of a quotient with a transition at a specific transition opportunity, given that it does not violate WRDS bounds, is a very simple mechanism that has been found to offer a compromise between optimization of the amount of transmitted timing information and the resulting occurrence of long like bit sequences. An explanation for this compromise is straightforward. Although this mechanism will not always select the quotient with the greatest number of transitions, it does ensure a lower bound to the number of transitions transmitted. When quotients are known to vary by only a few transitions, this selection mechanism can result in only a slight reduction in encoded stream transition density over that of the stream generated through selection of the quotient with the most transitions. Additionally, if the transition opportunity on which selection is based is appropriately chosen, the maximum possible length of encoded like bits can be reduced.

This mechanism can also be used in conjuction with other quotient selection mechanisms. For instance, if the quotient with the most transitions is to be selected and two or more quotients contain an equal number of transitions, selection of the quotient with a transition at an appropriately chosen transition opportunity from this quotient subset can further limit the maximum length of like encoded bits.

4.4 Shift Register Update

In the description of the Guided Scrambling concept, an example depicting accurate restoration of source sequences through self-synchronous scrambling decoding was given. The sequences, which differed only in their most significant bit, were recovered from very different transmitted bit streams using identical multiplication shift registers. In the same manner, Guided Scrambling decoding must recover the original source words regardless of the value of augmenting bit.

Guided Scrambling, however, involves the decoding of a continuous stream of codewords where each word has been selected from a set of quotients. Due to this quotient selection, continuous operation of both the division and multiplication registers will not ensure accurate restoration of the original bit stream. One or both of the encoding and decoding mechanisms must recognize the block nature of the fixed augmenting bit position code.

Three implementation efficient encoding and decoding configurations offer satisfactory performance:

(1) Both the encoder and decoder shift registers are cleared after every word resulting in a word by word operation and interpretation as a true block code.
(2) The encoding registers are updated after every word to create a transmitted stream which can be decoded through continuous multiplication.
(3) A combination of (1) and (2) given the appropriate n_a and d(x).

Each alternative is considered below.

Block encoding/decoding:

As shown in Appendix 1, the mapping between augmented source words and quotients is one-to-one. The division and multiplication processes of GS encoding and decoding perform this mapping correctly as long as all registers are cleared prior to the consideration of each word.

A diagram of a Block Guided Scrambling encoder and decoder when augmentation is with a single bit per word is given in Figure 4.3.

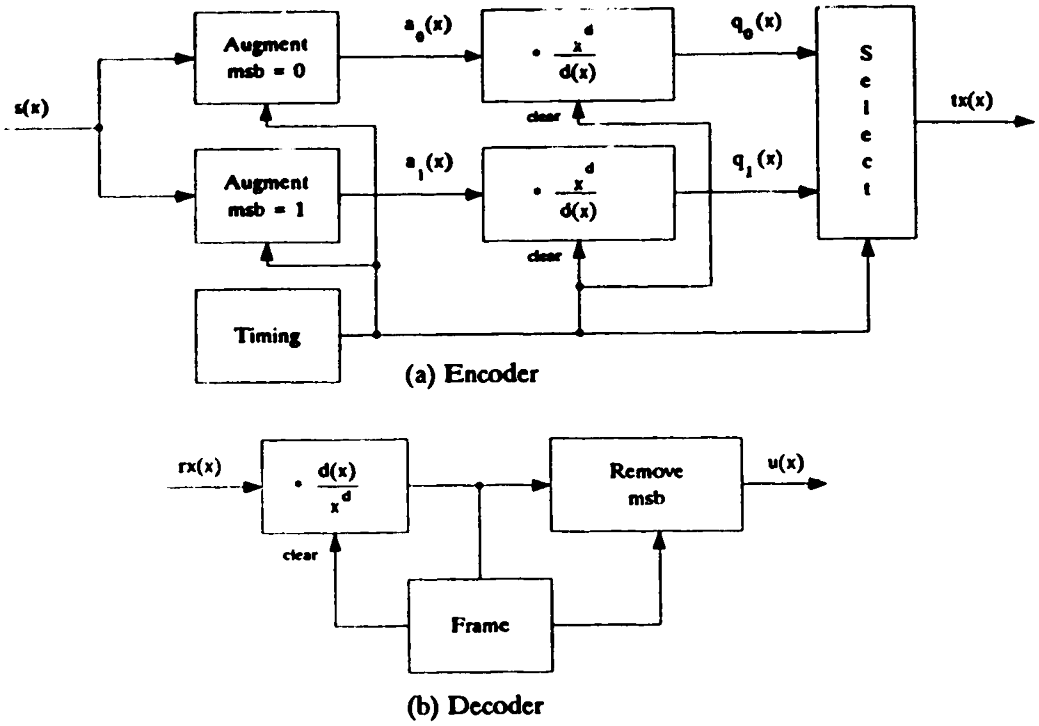


Figure 4.3: Block Guided Scrambling, $n_a = 1$.

Continuous encoding/decoding:

Shift registers are memory devices. The continuous division and multiplication operations of self-synchronizing scrambling are made possible through retention of sequence history information in shift register flip-flops. If Guided Scrambling is to decode through continuous multiplication, the received bit stream must appear as if encoder division has been continuous. But selection from a set of shift register outputs during encoding dictates that the transmitted stream will not be the continuous output from a single shift register. However, the transmitted stream can be easily made to appear as if it is the output from a single shift register.

Following source word augmentation and division, each of the encoder division registers contains the remainder of the division by d(x) of a different $a_i(x)$. The correct stream history information will be in the register which generated the $q_i(x)$ chosen for transmission. If all other division registers are updated to contain this remainder immediately following quotient selection, each quotient of the subsequent selection set will appear to result from continuous division. This update can be performed through either a direct register load or modification by the known difference in remainders. Continuous encoder register updating in this manner allows for continuous multiplication at the decoder.

A block diagram depicting the generation and decoding of a continuous bit stream with one augmenting bit per word is given in Figure 4.4.

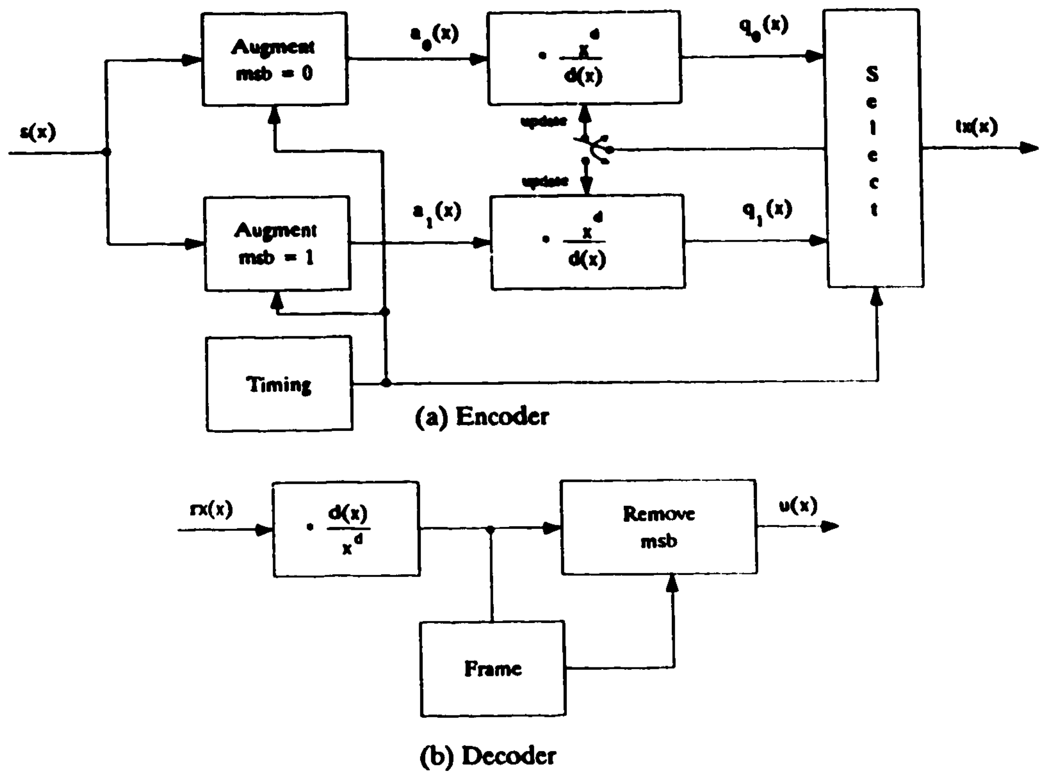


Figure 4.4: Continuous Guided Scrambling, $n_a = 1$.

Combined Block/Continuous structure:

Consider the operation of encoding division registers when the degree of the scrambling polynomial is less than or equal to the number of augmenting bits. If the encoder division registers are cleared after every word, augmented bit streams enter their respective shift registers without any alteration to the augmenting bits. But as indicated in Appendix 1, when these registers contain the remainder of previous division, an effective modulo-2 addition of the register contents with the most significant d bits of the augmented sequence occurs. If d $\leq n_a$, only augmenting bit positions are affected by this addition. When the registers have been updated to contain the same remainder, the leading bits of each augmented sequence undergo identical modulo-2 addition. If all 2^{n_a} augmenting bit combinations were originally inserted into the source stream, the same 2^{n_a} augmented bit streams result and the collective output of the division registers form the same selection set which would have been generated through block encoding.

To demonstrate, consider block encoding with $m=8$, $n_a=2$, $d(x)=x^2+1$. With a source word of 11010000, the augmented words and quotient selection set are:

$a_0(x)$	: 0011010000	$q_0(x)$	: 0011101010
$a_1(x)$	: 0111010000	$q_1(x)$	: 0110111111
$a_2(x)$	: 1011010000	$q_2(x)$	: 1001000000
$a_3(x)$	: 1111010000	$q_3(x)$	: 1100010101 .

If continuous encoding had resulted in a division register update with register values 10 prior to the consideration of this word, the effective augmented words $a_i'(x)$ and quotients become:

$a_0'(x)$	: 1011010000	$q_0(x)$	: 1001000000
$a_1'(x)$	: 1111010000	$q_1(x)$	: 1100010101
$a_2'(x)$	: 0011010000	$q_2(x)$	: 0011101010
$a_3'(x)$	: 0111010000	$q_3(x)$	: 0110111111 .

It is obvious that the quotient selection set consists of the same four quotients. With the same encoder selection mechanism, the same quotient would be selected and the identical encoded bit stream formed.

At the decoder, identical decoded bit streams also result from continuous and block decoding when $d \leq n_a$. Since the multiplication register does not involve feedback, clearing the contents of the multiplication register through block decoding affects at most the d most significant bits of each word. Since $d \leq n_a$ and the n_a leading bits are removed, the expurgated block and continuously decoded sequences will be the same.

To continue with the above example, if the same source word occurs successively, selection of $q_0(x)$ is followed by $q_3(x)$ and transmission is errorless, the received bit stream would be:

$$rx(x) = \underline{00}11101010\underline{11}00010101 \ ,$$

where the underlined bits occupy the augmenting bit positions. If this stream is simultaneously multiplied by $d(x)=x^2+1$ and divided by x^2, the product becomes:

$$p(x) = \underline{00}11010000\underline{01}11010000 \ .$$

If the decoding register was cleared following the multiplication of the first word, the resultant stream would instead be:

$$p(x) = \underline{00}11010000\underline{11}11010000 \ .$$

Obviously, removal of the differing bits in the augmenting bit positions leaves the same correct decoded sequence.

Consequently, when the degree of d(x) is less than or equal to the number of augmenting bits in each word and the quotient selection set consists of 2^{n_a} quotients, the same bit stream will be transmitted for both Block or Continuous GS encoding. Similarly, in the absence of transmission errors, the same decoded stream will result regardless whether decoding is in a block or continuous manner. In consideration of circuit complexity, the slightly simpler block encoder of Figure 4.3(a) and continuous decoder of Figure 4.4(b) can be jointly implemented.

Chapter 5 discusses the performance of a number of Guided Scrambling configurations.

CHAPTER 5

GUIDED SCRAMBLING PERFORMANCE

This chapter reports the performance of Guided Scrambling with respect to line code criteria. The first section summarizes performance in terms of encoded stream WRDS bounds, digital sum variation, maximum consecutive like bits, and minimum transition density. Code efficiency and the error extension which can be expected during decoding are also noted. Characterization of Guided Scrambling performance through examination of the influence its encoding rules have on the power spectral density of the transmitted bit stream is given in Section 5.2. Simulations which agree with theoretical expectations are discussed in Section 5.3. A comparison of Guided Scrambling performance with that of conventional line code techniques is given in each section and is summarized in Section 5.4.

5.1 Code Performance

The following characteristics of bit streams encoded through Guided Scrambling can be evaluated analytically:

(1) Word-end running digital sum
(2) Digital sum variation
(3) Maximum consecutive like bits
(4) Minimum transition density
(5) Error extension
(6) Efficiency .

The analysis of these performance criteria is given in Appendix 2. This section summarizes analysis results.

Tables 5.1 through 5.6 present Guided Scrambling performance for configurations using the scrambling polynomials derived in the previous chapter. Performance is quoted for the quotient selection mechanisms which give the best line code characteristics. Appendix 2 details the selection of these mechanisms from those suggested in Section 4.3. Performance figures reported in these tables agree with simulation of Guided Scrambling encoding rules.

Although Tables 5.1 - 5.6 give values for each measure of performance, they do not provide an intuitive feel for Guided Scrambling behavior. Due to complexity in their expression, comparison of DSV and L_{max} values with those for an alphabetic 7B8B code and the family of Partially Flipped codes given in Table 5.7 also offers little insight into advantages one technique might have over another. Perhaps a better grasp of GS code performance is available through the graphical comparison of DSV and L_{max} values in Figures 5.1 (a) - (d). These figures compare the performance of conventional line codes to that of Guided Scrambling with different families of scrambling polynomials. Each point or curve is labelled with code efficiency. Enforcement of different encoded stream running digital sum bounds results in a number of operating points of GS codes of the same efficiency. The closer the operating point is to the origin, the lower the encoded stream DSV and maximum number of consecutive like bits, and the better the line code performance.

Several conclusions regarding Guided Scrambling performance can be drawn from these tables and figures:

(1) Guided Scrambling cannot match the tight DSV and L_{max} bounds of the Alphabetic 7B8B code, but can provide slightly better performance than Partially Flipped codes.

(2) When augmentation is with one or two augmenting bits per word, transmission of the highest average number of level transitions is ensured when $d(x) = x^2+1$. As noted previously however, Guided Scrambling with a single augmenting bit and this scrambling polynomial cannot guarantee balanced transmission.

(3) Although the bound given for encoded stream transition density might in some cases appear low, the actual transition density can be much higher. Simulations discussed in Section 5.3 demonstrate that in most cases, transmission of much more timing information than these lower bounds indicate can be expected.

(4) Codes with scrambling polynomials of low weight give the best performance in terms of error extension among Guided Scrambling configurations. In the majority of configurations, truncation of errors during block decoding yields better performance than the corresponding continuous code. As shown in Appendix 2, the approximate error extension expressions can be taken as exact below bit error rates of 10^{-2}. Simulations discussed in Section 5.3 also demonstrate that Guided Scrambling error extension is similar to that of conventional line code techniques.

n	Selection Mechanism	L_{max}	td_{min}	Approximate Error Extension	Efficiency
odd	Select the quotient with the greatest number of transitions. Include $to_{(n-1)}$ in transition count.	n	$\frac{1}{2} + \frac{1}{2n}$	Continuous decoding: 2	$\frac{n-1}{n}$
even	Select the quotient with the greatest number of transitions. Exclude $to_{(n-1)}$ from transition count.	n	$\frac{1}{2}$	Block decoding: $\frac{2n-3}{n-1}$	

Table 5.1: Guided Scrambling Performance Summary:
$n_a = 1,\ d(x) = x^2 + 1$

n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
odd	Select quotient with minimum |WRDS| . If transition counts are equal, select quotient with transition in $to_{(n-1)}$.	$c \pm n$	$(\frac{3n-1}{2})$	$(\frac{5n-5}{2})$	$> \frac{1}{L_{max}}$	Block or Continuous decoding: 2	$\frac{n-1}{n}$
	Select the quotient with transition in $to_{(n-1)}$, given enforcement of WRDS bounds.	$c \pm (n+k)$, $0 \leq k \leq (\frac{n-3}{2})$	$(\frac{3n-1}{2}) + k$, $0 \leq k \leq (\frac{n-3}{2})$	$(\frac{5n-5}{2}) - k$, $0 \leq k \leq (\frac{n-3}{2})$			
		$\geq c \pm \frac{3}{2}(n-1)$	$\geq (2n-2)$	$2n-1$			
even	Select quotient with minimum |WRDS| . If transition counts are equal, select quotient with transition in $to_{(n-1)}$.	$c \pm n$	$(\frac{3n}{2} - 1)$	$(\frac{5n}{2} - 3)$			
	Select the quotient with transition in $to_{(n-1)}$, given enforcement of WRDS bounds.	$c \pm (n+k)$, $0 \leq k \leq (\frac{n}{2} - 2)$	$(\frac{3n}{2} - 1) + k$, $0 \leq k \leq (\frac{n}{2} - 2)$	$(\frac{5n}{2} - 3) - k$, $0 \leq k \leq (\frac{n}{2} - 2)$			
		$\geq c \pm (\frac{3n}{2} - 2)$	$\geq (2n-3)$	$2n-1$			

Table 5.2: Guided Scrambling Performance Summary:
$n_a = 1$, $d(x) = x + 1$

n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
even	Select quotient with minimum |WRDS|. If transition counts are equal, select quotient with transition in $to_{(n-1)}$.	$c \pm (n-2)$	$(\frac{3n}{2} - 2)$	$(\frac{5n}{2} - 2)$	$> \frac{1}{L_{max}}$	Continuous decoding: $[(\frac{n}{2} + 1) + Int(\frac{n-4}{6})]$ Block decoding: $(\frac{1}{n-1})\left[-1 + \sum_{i=1}^{R} i\,[Int(\frac{i+3}{2}) - Int(\frac{i}{2})]\right]$ $R = \begin{cases} \frac{2n}{3}, & n \bmod 6 = 0 \\ \frac{2n-1}{3}, & n \bmod 6 = 2 \\ \frac{2n+1}{3}, & n \bmod 6 = 4 \end{cases}$	$\frac{n-1}{n}$
	Select the quotient with the most transitions, given enforcement of WRDS bounds. Need consider first three transition opportunities only.	$c \pm (n-2+k)$, $0 \le k \le (\frac{n}{2} - 1)$	$(\frac{3n}{2} - 2) + k$, $0 \le k \le (\frac{n}{2} - 1)$	$(\frac{5n}{2} - 2) - k$, $0 \le k \le (\frac{n}{2} - 1)$			
		$\ge c \pm (\frac{3n}{2} - 3)$	$\ge (2n - 3)$	$2n - 1$			
	Select the quotient with transition in $to_{(n-1)}$, given enforcement of WRDS bounds.	$c \pm (n-2+k)$, $0 \le k \le (\frac{n}{2} - 2)$	$(\frac{3n}{2} - 2) + k$, $0 \le k \le (\frac{n}{2} - 2)$	$(\frac{5n}{2} - 2) - k$, $0 \le k \le (\frac{n}{2} - 2)$			
		$\ge c \pm (\frac{3n}{2} - 4)$	$\ge (2n - 3)$	$2n - 2$			

Table 5.3: Guided Scrambling Performance Summary:
$n_a = 1,\ d(x) \in D_{mod6}(x)$

n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
even	Select quotient with minimum |WRDS| . If transition counts are equal, select quotient with transition in $to_{(n-1)}$.	$c \pm (n-2)$	$(\frac{3n}{2} - 3)$	$(\frac{5n}{2} - 3)$	$> \frac{1}{L_{max}}$	Continuous decoding: 3 Block decoding: $\frac{2n}{n-1}$	$\frac{n-1}{n}$
	Select the quotient with a transition in $to_{(n-1)}$, given enforcement of WRDS and DSV bounds.	$c \pm (n-2+k)$, $0 \le k \le (\frac{n}{2} - 1)$	$(\frac{3n}{2} - 3) + k$, $0 \le k \le (\frac{n}{2} - 1)$	$(\frac{5n}{2} - 3) - k$, $0 \le k \le (\frac{n}{2} - 1)$			
		$\ge c \pm (\frac{3n}{2} - 3)$	$\ge (2n-4)$	$2n-2$			

Table 5.4: Guided Scrambling Performance Summary:
$n_a = 1$, $d(x) = x^{n-2} + x^{n-3} + 1$

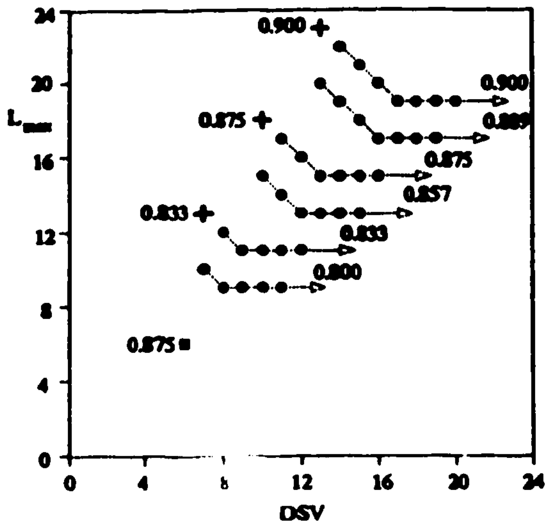
n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
even	Select quotient with minimum |WRDS| . If transition counts are equal, select quotient with transition in $to_{(n-1)}$.	$c \pm (n-2)$	$(\frac{3n}{2} - 3)$	$(\frac{5n}{2} - 3)$	$> \frac{1}{L_{max}}$	Continuous decoding: 3 Block decoding: $\frac{2n-1}{n-1}$	$\frac{n-1}{n}$
	Select the quotient with a transition in $to_{(n-1)}$, given enforcement of WRDS and DSV bounds.	$c \pm (n-2+k)$, $0 \leq k \leq (\frac{n}{2} - 1)$	$(\frac{3n}{2} - 3) + k$, $0 \leq k \leq (\frac{n}{2} - 1)$	$(\frac{5n}{2} - 3) - k$, $0 \leq k \leq (\frac{n}{2} - 1)$			
		$\geq c \pm (\frac{3n}{2} - 3)$	$\geq (2n-4)$	$2n-2$			

Table 5.5: Guided Scrambling Performance Summary: $n_s = 1,\ d(x) = x^{n-1} + x^{n-2} + 1$

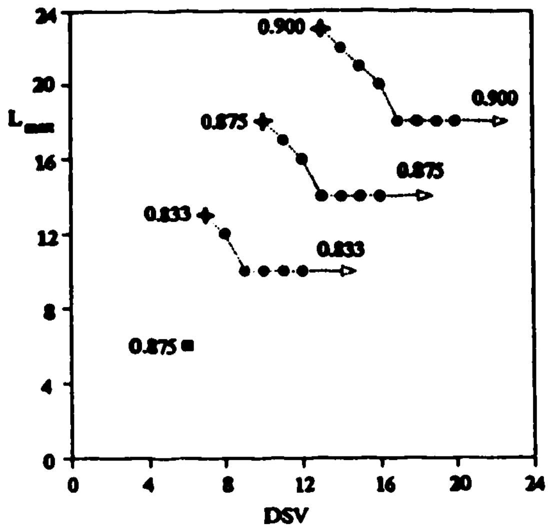
n	Secondary Selection*	WRDS bounds	DSV bound	L_{max}	td_{min}	Error Extension	Efficiency
n mod 4 = 0	Select quotient with transition in to_i, $(n-1) \geq i \geq (\frac{n}{2}-1)$	$\frac{n}{2}$	$\frac{3n-4}{4}$	$n-2$	$> \frac{1}{2}$	Block or Continuous decoding: 2	$\frac{n-2}{n}$
n mod 4 = 1	Select quotient with transition in to_i, $(n-2) \geq i \geq (\frac{n+1}{2})$	$\frac{n+1}{2}$	$\frac{3n+1}{4}$	$4,\ n = 5$ $n-2,\ n > 5$	$> (\frac{1}{2} - \frac{1}{2n})$		
n mod 4 = 2	Select quotient with transition in any predetermined transition opportunity.	$\frac{n-2}{2}$	$\frac{3n-10}{4}$	$8,\ n = 10$ $n-1,\ n > 10$	$> (\frac{1}{2} - \frac{1}{n})$		
n mod 4 = 3	Select quotient with transition in to_i, $(n-2) \geq i \geq 0$	$\frac{n-1}{2}$	$\frac{3n-5}{4}$	$n-1$	$> (\frac{1}{2} - \frac{1}{2n})$		

*Primary Selection Mechanism: Select quotient with most transitions that does not violate WRDS or DSV bounds. Inlude $tq_{(n-1)}$ in transition count.

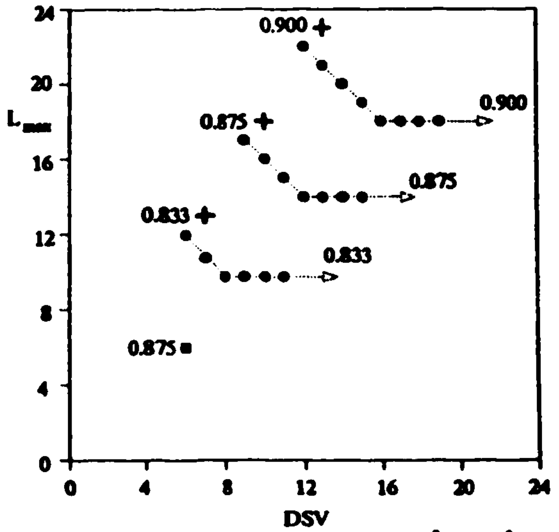
Table 5.6: Guided Scrambling Performance Summary:
$n_a = 2,\ d(x) = x^2 + 1$



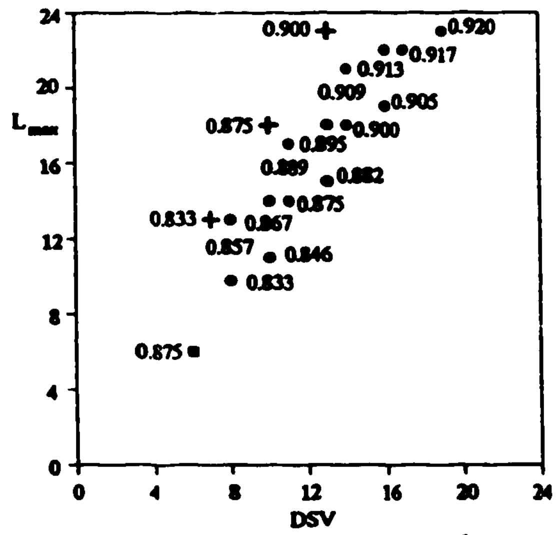
(a) GS code. $n_s = 1$, $d(x) = x + 1$.



(b) GS code: $n_s = 1$, $d(x) \in D_{most}(x)$



(c) GS code: $n_s = 1$, $d(x) = x^{n-2} + x^{n-3} + 1$
o. $d(x) = x^{n-1} + x^{n-2} + 1$



(d) GS code: $n_s = 2$, $d(x) = x^2 + 1$.

■ Alphabetic 7B8B ✚ PFmB(m+1)B • Guided Scrambling

Points/Curves labelled with efficiency.

Guided Scrambling selection mechanisms:

$n_s = 1$: Select quotient with preceding transition, given enforcement of WRDS bounds.
$n_s = 2$: See Table 5.6.

Figure 5.1: Comparison of Maximum Consecutive Like Bits vs. Digital Sum Variation

Code	DSV	L_{max}
7B8B	6	6
PFmB(m+1)B, m odd	$\frac{3m-1}{2}$	$\frac{5m+1}{2}$

Table 5.7: DSV and L_{max} for the Alphabetic 7B8B and Partially Flipped family of block codes.

5.2 Spectra of Encoded Signals

Another way of characterizing line code performance is by evaluating the influence that the encoding rules have on the power spectral density of the transmitted bit stream. Cariolaro and Tronca [24] outline an efficient procedure for calculating the spectra of balanced block codes. Details of this calculation are given in Appendix 3. In brief, the power spectral density of the cyclostationary transmitted bit stream can be evaluated by taking the Fourier transform of its autocorrelation. By separating the continuous line code spectral component $X_c(f)$ from the discrete components (with weighting function X_d), the power spectral density $\Phi(f)$ can be expressed as:

$$\Phi(f) = f_n \, |S(f)|^2 \, \{ X_c(f) + f_n X_d(f) \sum_{k=-\infty}^{\infty} \delta(f - kf_n) \} \, , \tag{5.1}$$

where:

$S(f)$	=	the Fourier transform of the transmitted symbol shape
f_n	=	$^1/_{nT}$
T	=	duration of each symbol
n	=	length of encoded word ,

and $X_c(f)$ and $X_d(f)$ depend only on the encoding rules of the line code. Calculation of $X_c(f)$ and $X_d(f)$ then gives the influence that encoding rules have on the spectra of the encoded bit stream independent of the pulse shape used for symbol transmission. When the transmitted pulse is assumed to be the delta function, $X_c(f)$ and $X_d(f)$ completely characterize the power spectral density of the encoded signal. When $S(f)$ is a function other than unity, $X_c(f)$ can be interpreted as a function which filters the spectrum of the transmitted pulse.

Figure 5.2 gives the continuous spectral component of an alphabetic 7B8B block code described with code tables by Sharland and Stevenson [10] and the Partially Flipped 7B8B code. Because these spectra are symmetric and periodic, only the normalized frequency range $0 \leq fT \leq 1.0$ (T the duration of each symbol) is given. That the spectra for the alphabetic code vary with input bit stream logic one probability (p) is expected due to the dependency of input word occurrence and thus the codewords transmitted on source stream dc level. The similarity of Partially Flipped code spectra for high and low values of source stream logic one probability is also expected due to encoding rules which result in transmission of either the source word or its complement.

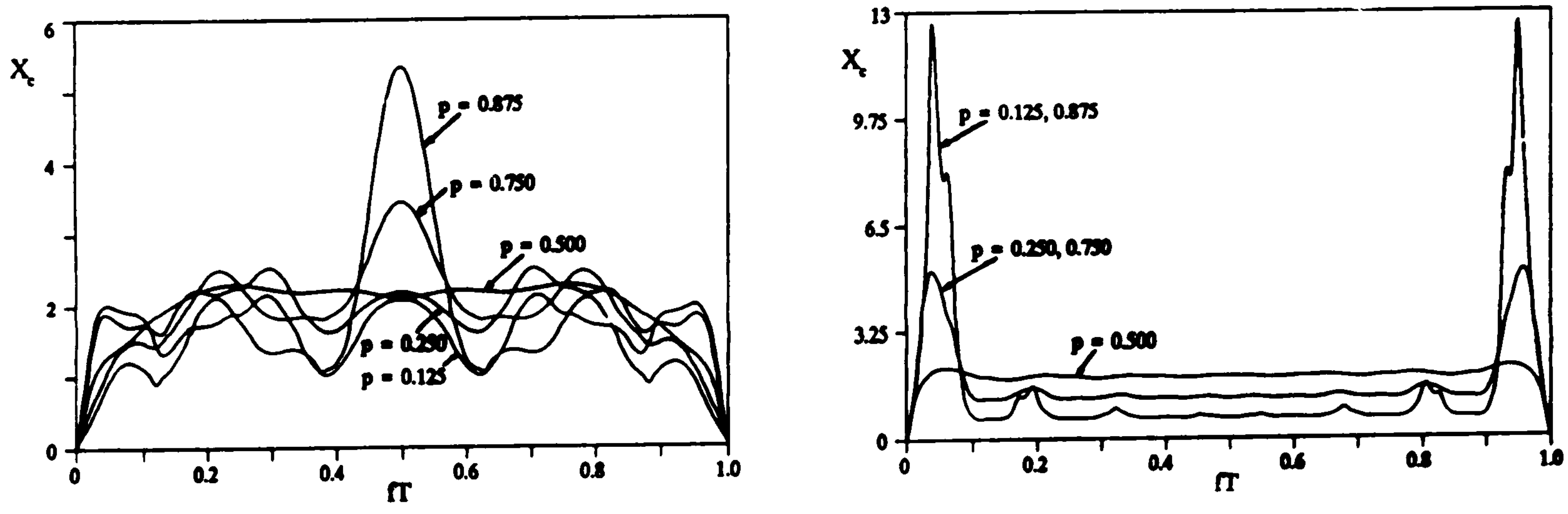
Figures 5.3 through 5.7 depict the continuous spectral components of a number of Guided Scrambling configurations. Relatively short GS codes were considered in order to limit computational complexity and to allow for comparison with the established alphabetic code of similar efficiency. The following general statements can be made regarding the spectra of these line codes:

(1) Balanced transmission is characterized by a continuous spectral component which falls to zero as frequency approaches zero. The conventional line codes and all balanced GS configurations considered exhibit this characteristic.

(2) The greater the power at low frequencies, the greater the expected low frequency wander. This spectral characteristic may correspond to such poor line code characteristics as the transmission of many long like bit sequences or a slowly modulated encoded stream running digital sum. The Partially Flipped code exhibits large spectral peaks at low frequencies for both high and low values of source stream logic one probability. Block GS codes exhibit slightly smaller spectral peaks at these frequencies for low values of source stream logic one probability. Guided Scrambling configurations which select the word with minimum |WRDS| contain less power very close to dc than GS configurations with other selection mechanisms, but also exhibit larger spectral peaks at low frequencies than codes configured with other selection mechanisms. Continuous GS codes are not much inferior to the alphabetic line code with respect to the amount of power transmitted at low frequencies.

(3) Codes which contain a large spectral peaks near $fT = 1$ require high order filtering if spectral information directly above this frequency must be discarded. This is cause for concern in frequency division multiplexed systems which attempt to restrict transmission to as narrow a frequency band as possible. However, high bit rate fiber systems typically employ high pass filtering only to reduce received noise. Since the order of these filters is usually low, large spectral components near $fT = 1$ are of little concern. Symmetry of code spectra about $fT = 0.5$ indicates that observations of (2) concerning code comparison apply also to this criterion.

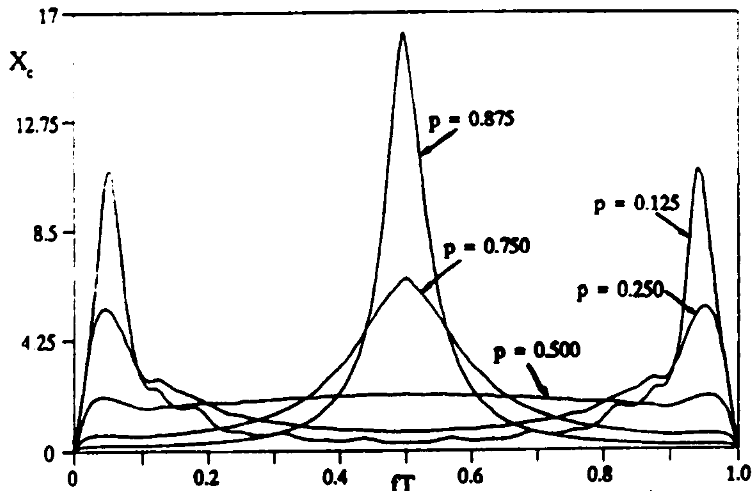
(4) Spectra with large components near $fT = 0.5$ characterize line codes that ensure transmission of bit sequences with a high average number of level transitions. This correspondence becomes clear when it is observed that the alternating sequence which contains the greatest number of transitions has a fundamental frequency of $f = {}^{1}/_{2T}$. Figure 5.8 demonstrates this relationship. Block Guided Scrambling codes for high source sequence logic one probability contain the largest spectral components near $fT = 0.5$. Many are superior to the alphabetic line code in this respect.



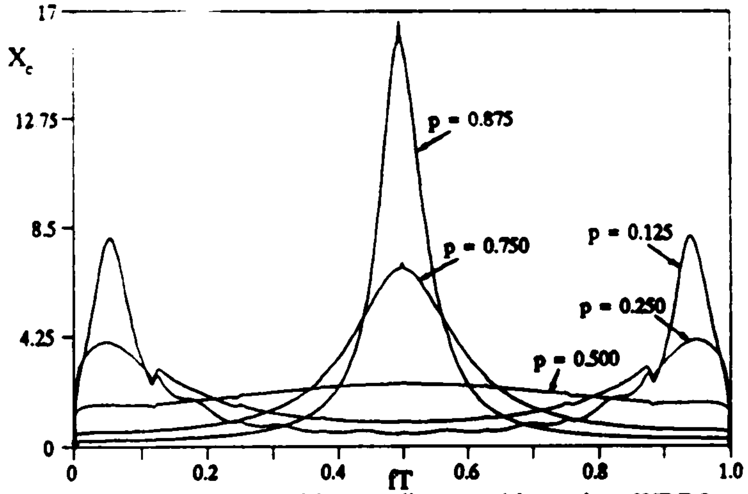
(a) Alphabetic 7B8B

(b) PF7B8B

Figure 5.2: Power Spectral Density: Alphabetic 7B8B and PF7B8B Codes.

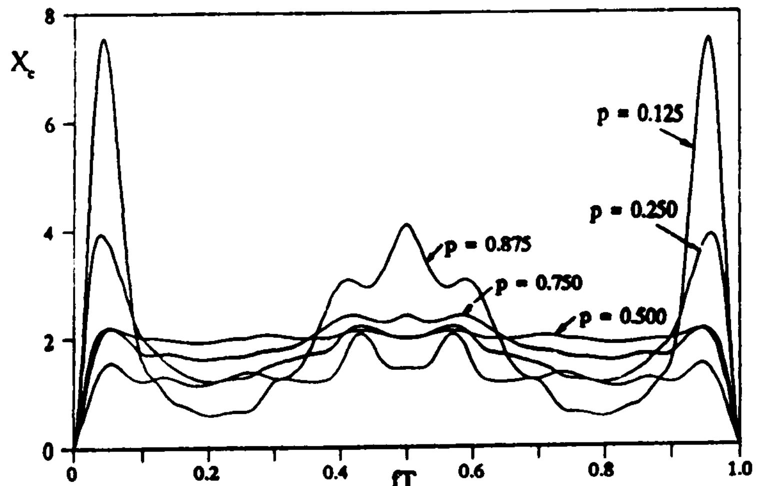


(a) Select quotient with minimum |WRDS|

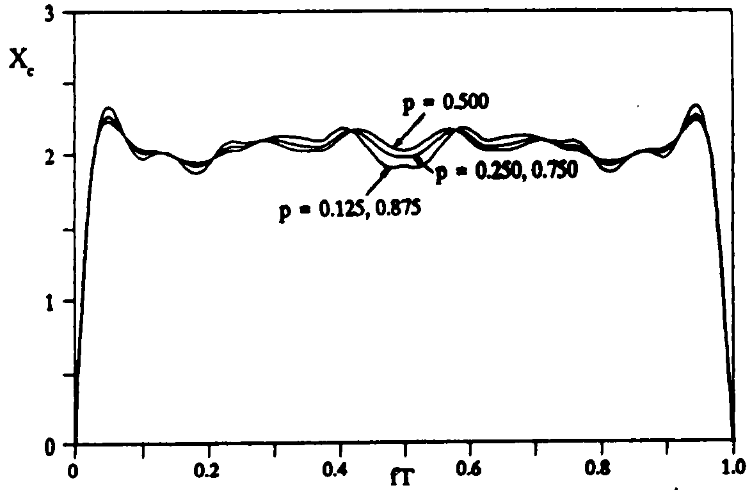


(b) Select quotient with preceding transition, given WRDS bounds of ±8

Figure 5.3: GS7B8B Power Spectral Density: d(x) = x + 1.

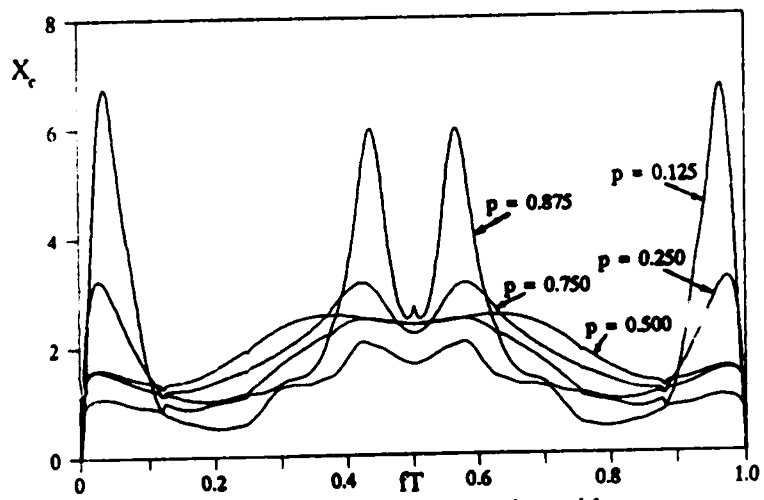


(a) Block Encoding: Select quotient with minimum |WRDS|

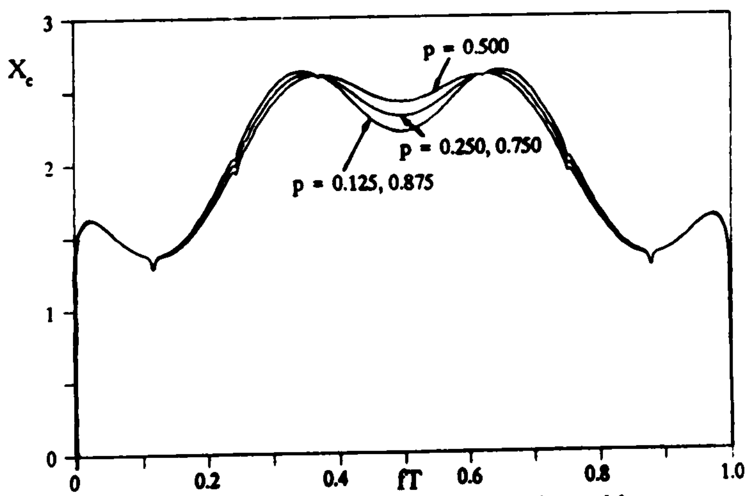


(b) Continuous Encoding: Select quotient with minimum |WRDS|

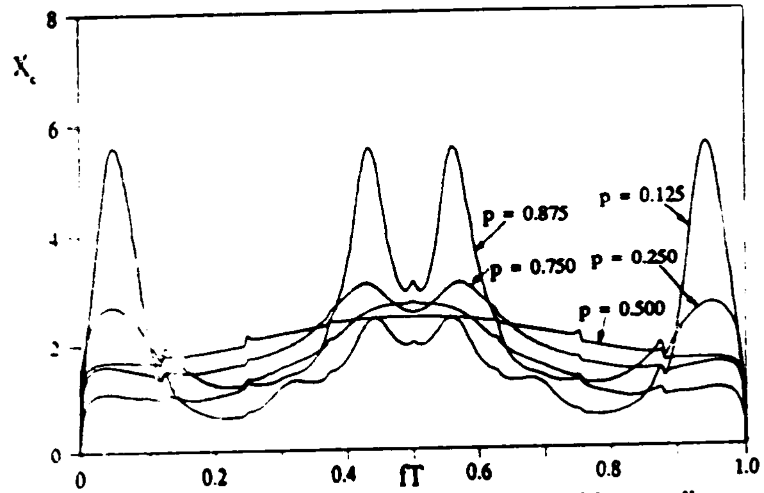
Figure 5.4: GS7B8B Power Spectral Density: $d(x) = x^6 + x^4 + x^3 + x + 1$.



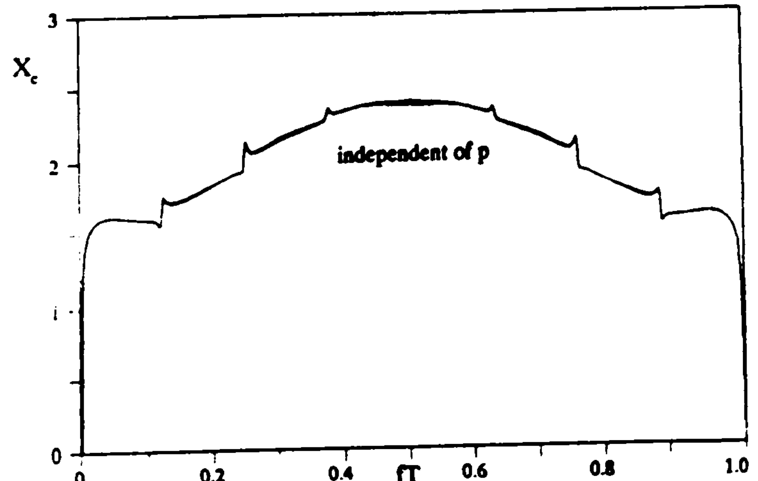
(c) Block Encoding: Select quotient with most transitions, given WRDS bounds of +10/-8



(d) Continuous Encoding: Select quotient with most transitions, given WRDS bounds of +10/-8

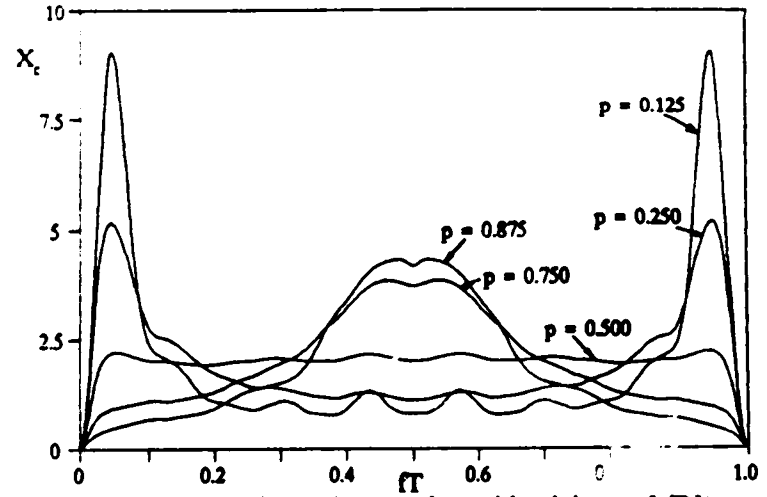


(e) Block Encoding: Select quotient with preceding transition, given WRDS bounds of +10/-8

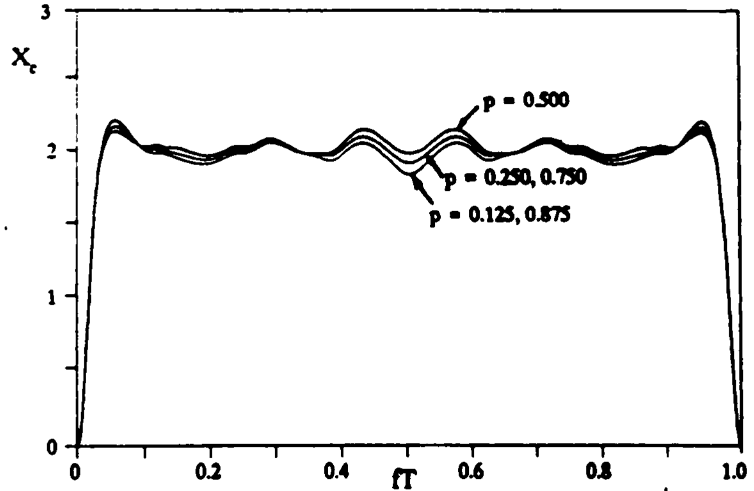


(f) Continuous Encoding: Select quotient with preceding transition, given WRDS bounds of +10/-8

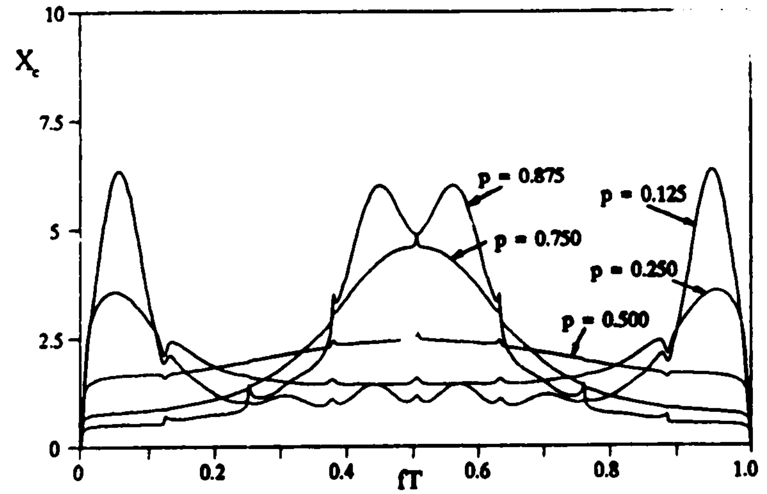
Figure 5.4: GS7B8B Power Spectral Density: $d(x) = x^6 + x^4 + x^3 + x + 1$, continued.



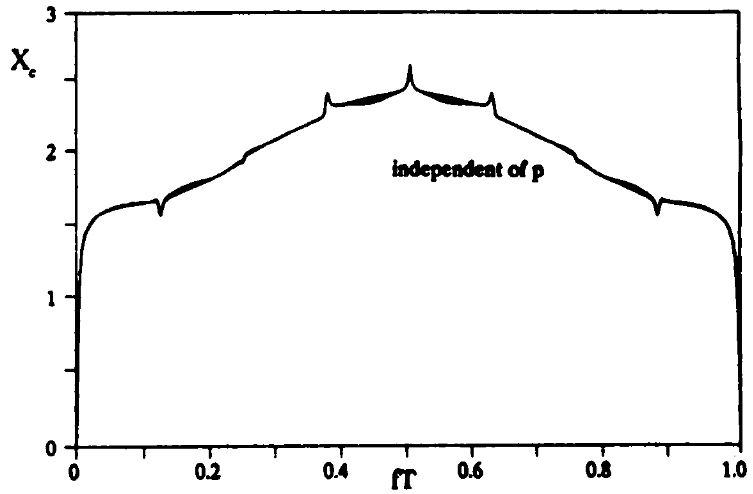
(a) Block Encoding: Select quotient with minimum |WRDS|



(b) Continuous Encoding: Select quotient with minimum| WRDS|

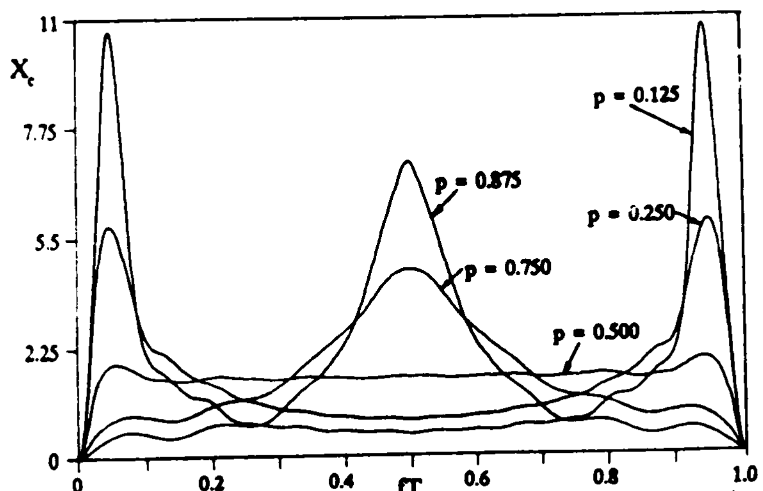


(c) Block Encoding: Select quotient with preceding transition, given WRDS bounds of +10/-8, RDS bounds of +13/-11.

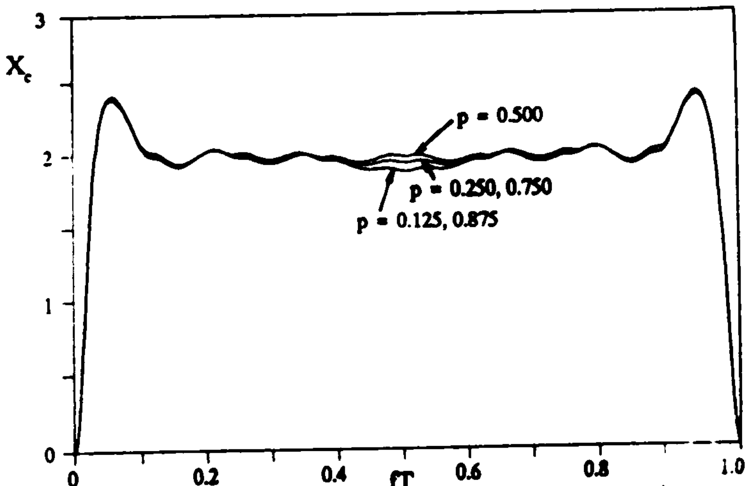


(d) Continuous Encoding: Select quotient with preceding transition, given WRDS bounds of +10/-8, RDS bounds of +13/-11.

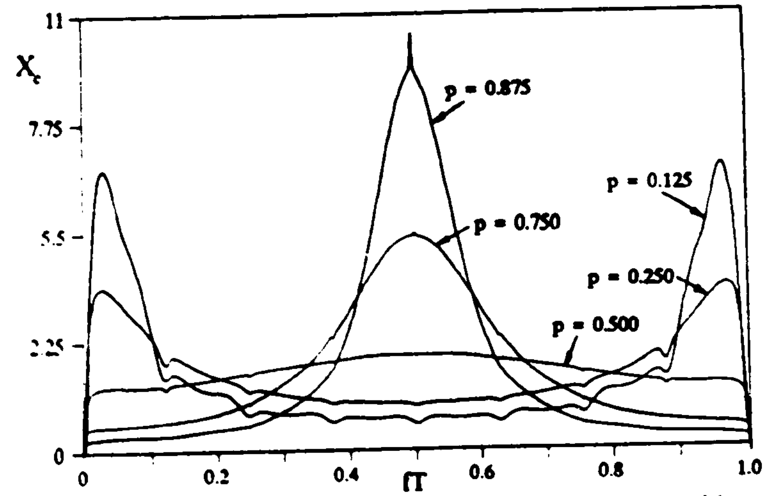
Figure 5.5: GS7B8B Power Spectral Density: $d(x) = x^6 + x^5 + 1$



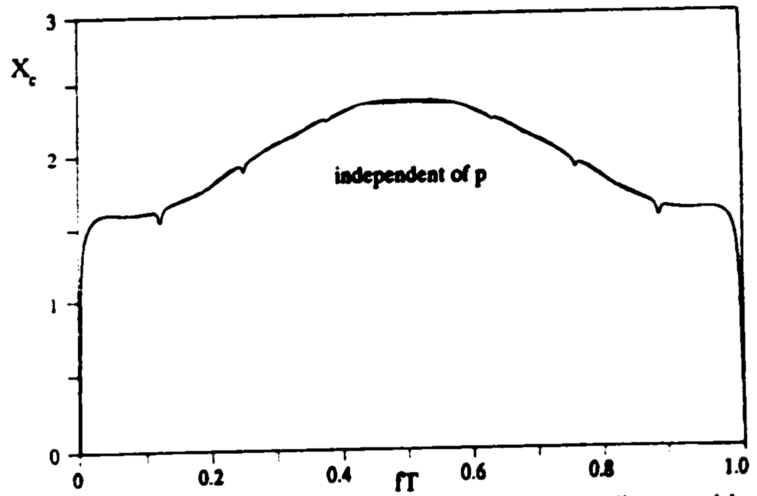
(a) Block Encoding: Select quotient with minimum| WRDS|



(b) Continuous Encoding: Select quotient with minimum| WRDS|

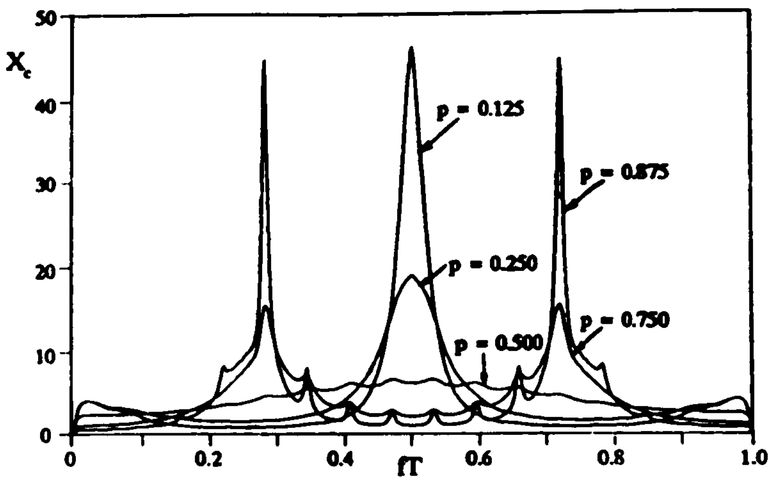


(c) Block Encoding: Select quotient with preceding transition, given WRDS bounds of +10/-8, RDS bounds of +13/-11.



(d) Continuous Encoding: Select quotient with preceding transition, given WRDS bounds of +10/-8, RDS bounds of +13/-11.

Figure 5.6: GS7B8B Power Spectral Density: $d(x) = x^7 + x^6 + 1$.



Select quotient with most transitions, given WRDS bounds of ± 8, RDS bounds of ± 11.

Figure 5.7: GS14B16B Power Spectral Density: $d(x) = x^2 + 1$.

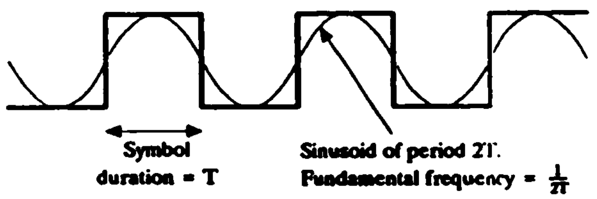


Figure 5.8: Fundamental Frequency of the Alternating Sequence

(5) The Alphabetic 7B8B, PF7B8B and Block GS codes display spectra which vary widely with input sequence logic one probability. But when the degree of the scrambling polynomial approaches the length of the encoded words, Continuous Guided Scrambling results in spectra which are almost independent of input sequence statistics. Independence of the transmitted signal spectrum from the statistics of the source bit stream simplifies system design.

To explain the spectral characteristics of GS codes, note first that the spectra of single augmenting bit Block GS codes indicate that these encoding rules result in transmission of more level transitions for high levels of source stream logic one probability than low levels. Since block codes encode on a word by word basis, it is expected that source words composed mostly of ones will be encoded to words with more transitions than source words consisting predominantly of zeros. Augmentation and division of source words prior to codeword selection result in exactly this situation.

Consider the Block Guided Scrambling encoding process when $n_a = 1$ and $d(x) = x+1$. If the input word consists entirely of zeros, the quotient selection set becomes:

$$
\begin{aligned}
q_0(x) &= 0000 \ldots 00 \\
q_1(x) &= 1111 \ldots 11 ,
\end{aligned}
$$

and a quotient with at most a single preceding transition can be selected. But when the input word is composed entirely of ones, the quotient selection set is:

$$
\begin{aligned}
q_0(x) &= 0101 \ldots 01 \\
q_1(x) &= 1010 \ldots 10 ,
\end{aligned}
$$

and an alternating sequence containing many more transitions can be selected. Other single augmenting bit GS codes exhibit similar properties. When augmentation is with two bits, it is straightforward to show that Guided Scrambling with $d(x) = x^2+1$ generates quotients containing more transitions when source stream logic one probability is low than when it is high.

An explanation for the near independence of Continuous Guided Scrambling spectra from input sequence logic one probability is also straightforward. To allow the source bit stream to be continuously encoded rather than encoded on a word by word basis, encoder division registers are updated rather than cleared following quotient selection. This register updating introduces feedback which effectively modifies the incoming source word with the remainder of previous division. Modification of the source stream in this manner alters its apparent logic one probability and so tends to remove the dependence of encoded stream characteristics from the source stream dc level.

Note that the amount of feedback introduced during shift register updating is related to the degree of the scrambling polynomial. The greater this degree, the greater the possible degree of remainders from division and the greater the number of incoming source bits which can be affected by register updating. Although the spectra of codes using 7th degree scrambling polynomials do not clearly demonstrate less dependence on source stream characteristics than those using polynomials of 6th degree, simulation of a Continuous GS code with a scrambling polynomial of only second degree demonstrates the retention of dependence on source stream properties. This simulation is discussed in the next section.

5.3 Line Code Simulation

Throughout the development of Guided Scrambling, computer simulation was used to verify the concept and confirm theoretical analysis. Section 5.3.1 gives a brief description of the environment created for simulation of Guided Scrambling and other line code encoding and decoding rules on the IBM PC family of computers. The simulations performed are outlined in Section 5.3.2.

The results of Guided Scrambling simulation agree with theoretical expectations. Code limitations given in Tables 5.1 through 5.6 were found to be valid and characteristics of the simulated codes agree with spectra reported in the previous section. Sections 5.3.3 through 5.3.5 discuss simulation agreement with theoretical code limitations and dependence of code performance on input sequence logic one probability. A brief discussion concerning the simulated error extension of Guided Scrambling and conventional line code techniques is given in Section 5.3.6.

5.3.1 Line Coding Environment

To provide easy access to the programs written for simulation and analysis of Guided Scrambling and conventional line coding techniques, a mouse or keyboard driven pull-down menu shell was constructed for the IBM PC family of machines. Selection of a menu entry allows for:

(1) generation of a source bit stream with user defined characteristics,
(2) encoding of the source bit sequence through reset or self-synchronizing scrambling, Alphabetic 7B8B or Partially Flipped encoding rules, and Guided Scrambling,
(3) placement of a number of different channel impairments on the transmitted bit stream,
(4) decoding with the applicable line code rules,
(5) analysis of the transmitted bit stream with respect to line code criteria and the decoded bit stream with respect to error extension, and
(6) line code spectral analysis.

The bit streams can be viewed at any point by listing the appropriate file or selecting the desired bit stream from the VIEW menu. A log of the simulations and analysis performed is automatically updated when programs are executed.

With the exception of the spectral analysis routine which is written in double-precision FORTRAN, all programs are written in Pascal and compiled with the Turbo Pascal 4.0 compiler. The complete Line Coding Environment is available from the Alberta Telecommunications Research Centre. The environment can be initialized by executing the LCE batch file which accompanies the environment programs or, for more convenient use, can be installed on a hard disk by running the included INSTALL program.

A more complete description of the simulation environment and its versatility is given in Appendix 4.

5.3.2 Extent of Simulation

To establish confidence in the Guided Scrambling performance figures given in Section 5.1, GS codes using polynomials from each family of suggested scrambling polynomials were simulated. Table 5.8 lists the simulated Guided Scrambling codes. These codes were configured with both block and continuous encoding and decoding rules and a number of quotient selection mechanisms. The majority of GS codes simulated were 87.5% efficient to allow for performance comparison with conventional line code techniques of similar efficiency. Simulation of the Alphabetic 7B8B and

GS Code	Polynomial	GS Code	Polynomial
7B8B	$x+1$ x^2+1 $x^6+x^4+x^3+x+1$ x^6+x^5+1 x^7+x^6+1	11B12B	$x^{11}+x^{10}+1$
		15B16B	$x^{15}+x^{14}+1$
8B9B	$x+1$ x^2+1	14B16B 15B17B 16B18B 17B19B	x^2+1

Table 5.8: Guided Scrambling Configurations Simulated

PF7B8B code rules provided this reference. The more efficient GS11B12B and GS15B16B codes were simulated to demonstrate changes in encoded stream characteristics as Guided Scrambling efficiency increases.

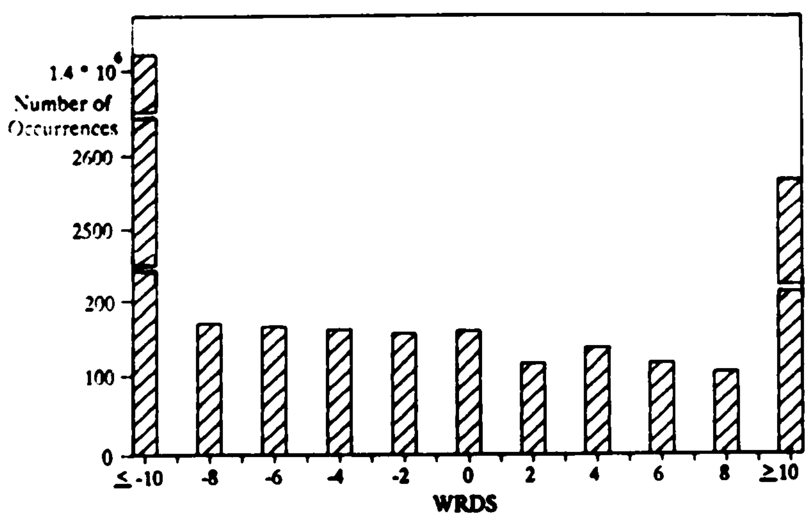
To simulate performance under a variety of conditions and to observe line code performance as a function of source stream dc, line coding rules were applied to nine sets of $1.008*10^6$ randomly generated source bits ranging in logic one probability from 0.1 through 0.9. Source sequences of this length were chosen since their length ensured statistically sound results, they could be transported from one machine to another on 1.2 Megabyte floppy disks, and they divided evenly into input words of all codes except the GS11B12B and GS17B19B code where 150 and 100 additional bits were included in each simulation respectively.

Analysis of each encoded bit sequence provided the line code performance data discussed in the next three sections. The inclusion of random bit errors into the encoded bit stream and subsequent decoding and error analysis allowed for examination of error extension exhibited by these line codes.

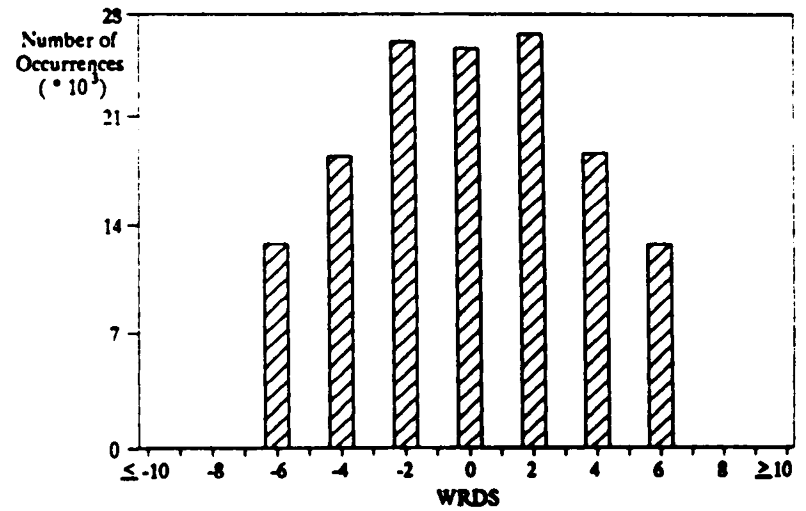
5.3.3 Encoded Stream WRDS and Digital Sum Variation

Straightforward computation of the running digital sum of bit sequences encoded through Guided Scrambling found the WRDS and DSV bounds for balanced GS codes to correspond to values given in Tables 5.2 through 5.6. Figure 5.9 depicts encoded stream WRDS distributions for a few Guided Scrambling configurations. Figure 5.10 gives similar distributions for the established line code techniques. Note that expected WRDS limits are demonstrated for the balanced codes. An unbounded WRDS for the GS code with $n_a = 1$, $d(x) = x^2+1$ characterizes its unbalanced nature.

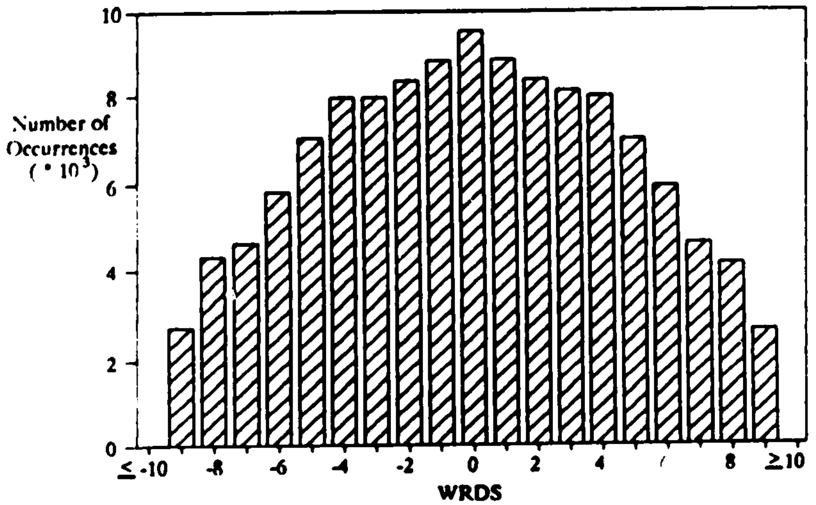
Table 5.9 gives representative DSV simulation results. As expected, enforcement of encoded stream WRDS and RDS bounds during Guided Scrambling ensures a bound on its digital sum variation.



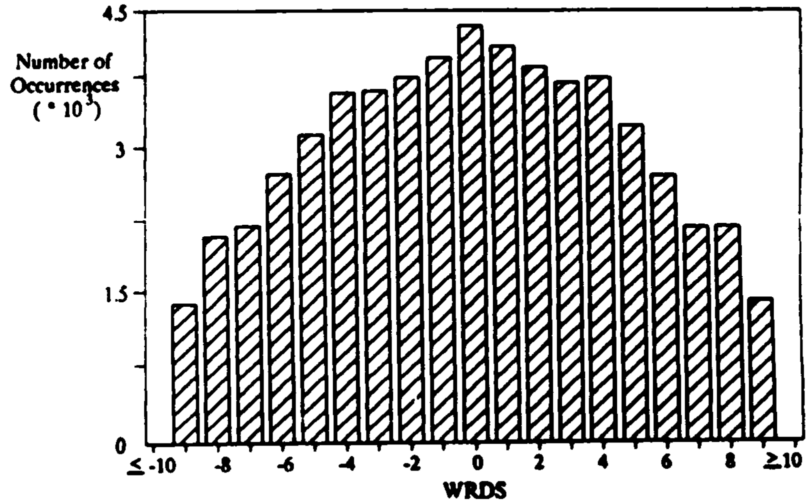
(a) GS7B8B, $d(x) = x^2 + 1$



(b) GS7B8B, $d(x) = x^7 + x^6 + 1$. WRDS bounds enforced: ± 6.

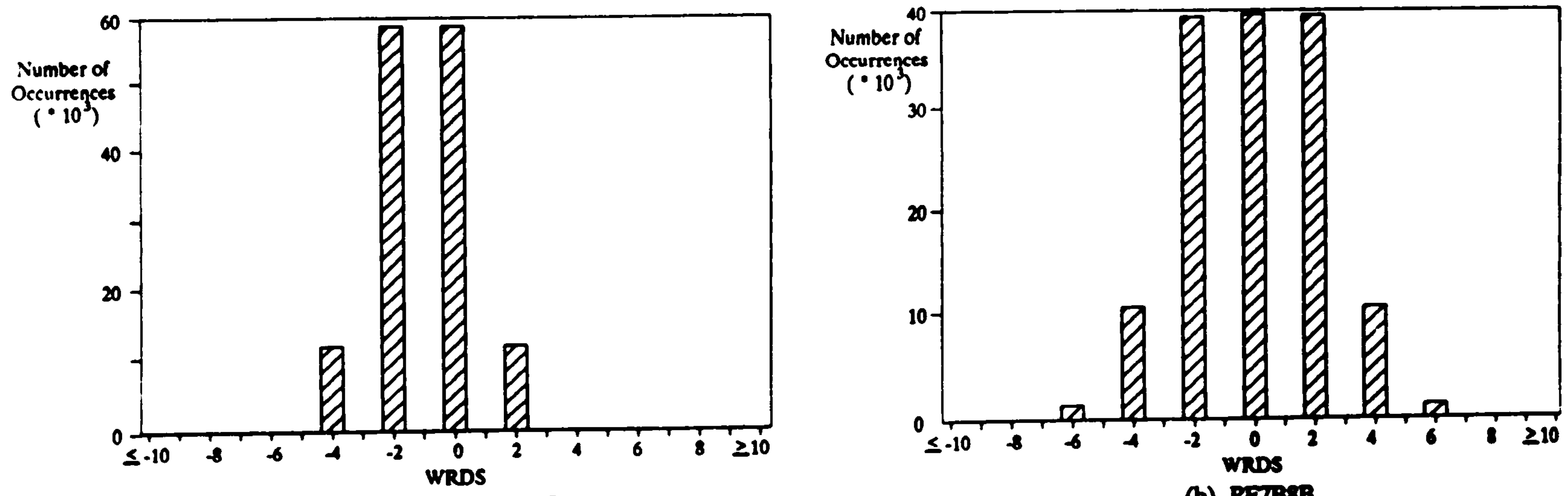


(c) GS8B9B, $d(x) = x + 1$. WRDS bounds enforced: ± 9.



(d) GS17B19B, $d(x) = x^2 + 1$. WRDS bounds enforced: ± 9.

Figure 5.9: Representative Guided Scrambling Encoded Stream WRDS Distributions



(a) Alphabetic 7B8B (b) PF7B8B

Figure 5.10: Conventional Line Code Encoded Stream WRDS Distributions

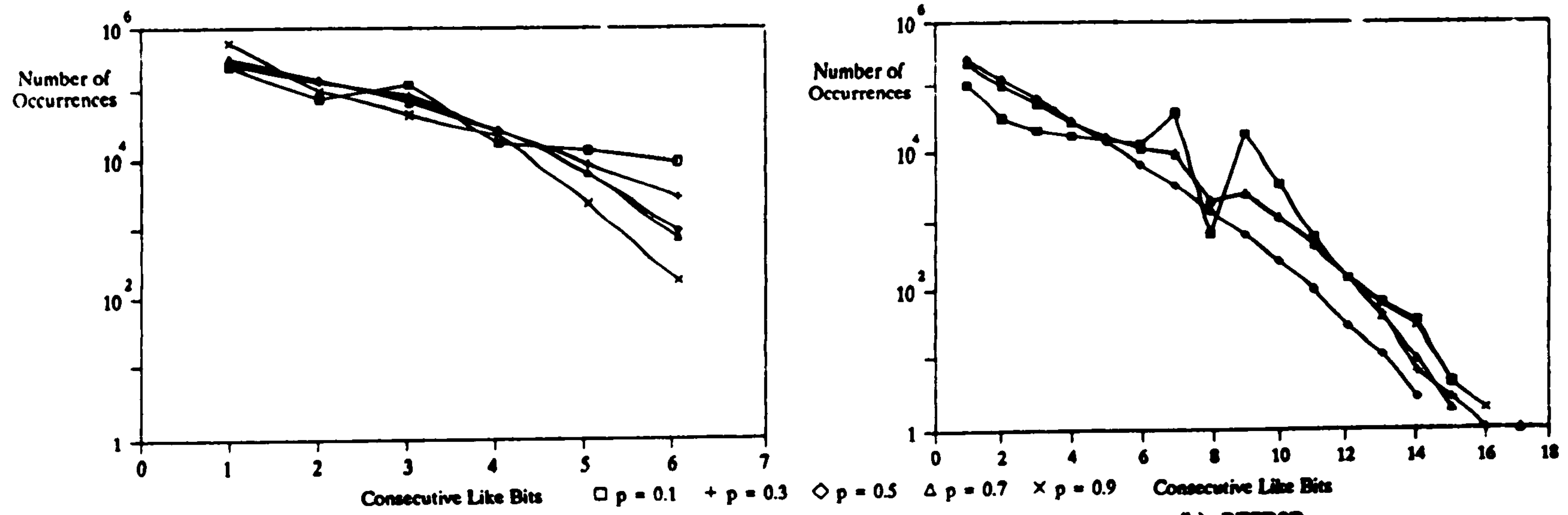
GS Code	WRDS Bounds Enforced	RDS Bounds Enforced	Expected DSV	Simulated DSV
7B8B $(x^7 + x^6 + 1)$	± 6 $+8/-6$ ± 8 $+10/-8$	± 9 $+11/-9$ ± 11 $+13/-11$	9 10 11 12	9 10 11 12
8B9B	± 9 ± 10 ± 11 ± 12	—	13 14 15 16	13 14 15 16
14B16B	± 8	± 11	11	11
15B17B	± 9	± 13	13	13
16B18B	± 8	± 11	11	11
17B19B	± 9	± 13	13	13

Table 5.9: Representative Guided Scrambling DSV Simulation Results

5.3.4 Consecutive Like Bit Distribution

Guided Scrambling simulation also found the maximum length of consecutive like bits in the encoded bit stream to be constrained as expected. But more interesting than validation of L_{max} constraints was the change in distribution of consecutive like bits that a number of line code configurations exhibited as source stream logic one probability was varied. Figure 5.11 gives encoded stream consecutive like bit distributions which resulted when the alphabetic and Partially Flipped encoding rules were applied to a number of source bit sequences. Consecutive like bit distributions for selected Guided Scrambling configurations are given in Figure 5.12. These distributions are representative of those evaluated for single and double augmenting bit block encoded GS codes and the Continuous GS codes with $n_a = 1, d > n_a$.

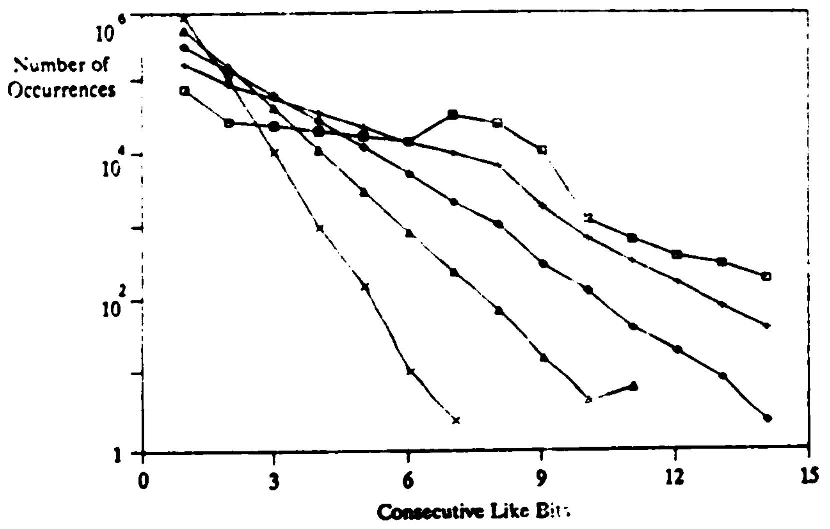
Changes in the distribution of consecutive like-bits indicate a change in the amount of timing information transmitted. That the distributions vary as simulation results demonstrate is expected



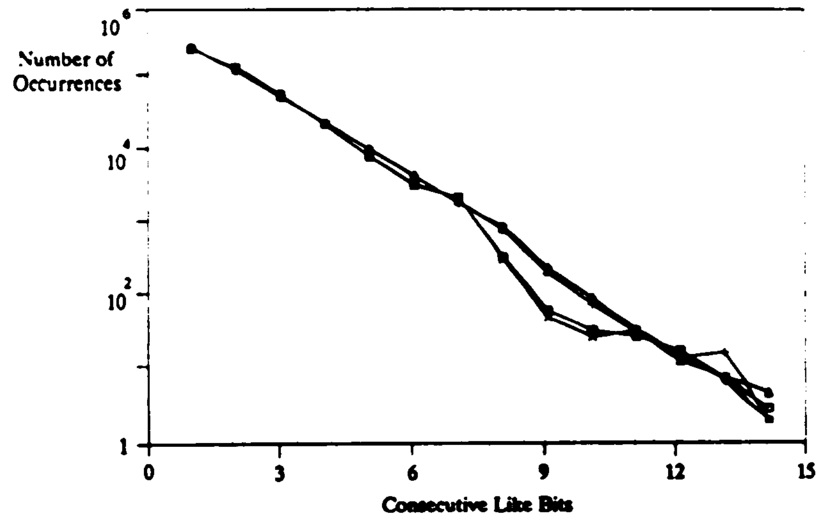
(a) Alphabetic 7B8B

(b) PF7B8B

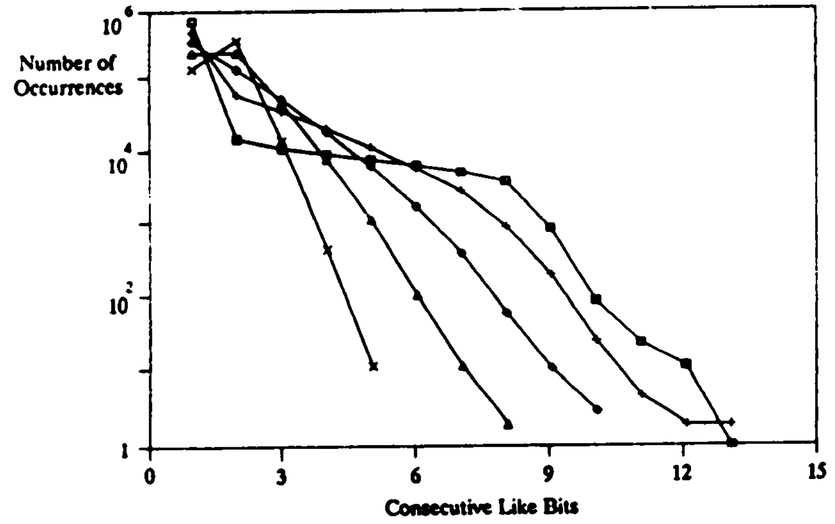
Figure 5.11: Conventional Line Code Consecutive Like Bit Distributions



(a) Block Encoded GS7B8B, $d(x) = x^7 + x^6 + 1$



(b) Continuously Encoded GS7B8B, $d(x) = x^7 + x^6 + 1$



(c) GS14B16B, $d(x) = x^2 + 1$

□ p = 0.1 + p = 0.3 ◇ p = 0.5 △ p = 0.7 × p = 0.9

Quotient Selection Mechanisms:

GS7B8B: Select quotient with preceding transition, given WRDS bounds of +10/-8, RDS bounds of +13/-11.

GS14B16B: Select quotient with most transitions, given WRDS bounds of ± 8, RDS bounds of ± 11.

Figure 5.12: Representative Guided Scrambling Consecutive Like Bit Distributions

when the code spectra of Figures 5.2 through 5.7 are considered:

(1) Spectra of the Alphabetic 7B8B code indicate that source streams with a high dc value will be encoded to contain more level transitions than source streams of a low dc level. The consecutive like bit distributions of Figure 5.11(a) demonstrate this property through an increase in the occurrence of single bits and a decrease in the occurrence of long like-bit sequences in the transmitted bit steam as the source stream logic one probability increases.

(2) Similarity between the Partially Flipped encoded consecutive like bit distributions for $p = 0.3$ and $p = 0.7$, and $p = 0.1$ and $p = 0.9$ agree with the similarity of spectra given in Figure 5.2(b). The spectra indicate and simulation results confirm that the greatest number of level transitions are transmitted when $p = 0.5$ and that this number is reduced symmetrically when source stream dc either increases or decreases from this value.

(3) As with the Alphabetic 7B8B code, spectra of single augmenting bit Block GS configurations indicate that the average number of level transitions transmitted increases with increasing source stream logic one probability. But the wide variation in Block GS code spectra also indicates that much more disparity in the number of level transitions transmitted should be found during Block Guided Scrambling than during encoding with the alphabetic look-up table codes. The consecutive like bit distributions of Figure 5.12(a) which demonstrate both wide variation and a decrease in expected number of consecutive like bits as source dc increases agree with these spectral characteristics.

(4) At first glance, the decrease in maximum consecutive like-bits in the GS14B16B encoded bit stream as source stream logic one probability increases appears to indicate that the number of level transitions should increase with source dc. But note the decrease of stand-alone bits and the increase of two consecutive like bits when source streams of increasing logic one probability are encoded. The increase in occurrence of two consecutive like bits in bit streams encoded from source streams of high logic one probability indicates that these streams should contain large spectral components at $f = {}^{1}/_{4T}$ while bit sequences encoded from source streams of low dc should contain larger spectral components at $f = {}^{1}/_{2T}$. Indeed, the spectra of Figure 5.7 demonstrate this property.

(5) Spectra of bit streams resulting from Continuous Guided Scrambling where the degree of the scrambling polynomial approaches the length of the codewords are almost independent of the dc level of the incoming source bit stream. The consecutive like bit distributions of Figure 5.12(b) demonstrate this independence, especially for the short consecutive bit length sequences where the majority of information is transmitted.

Perhaps these characteristics are more clearly demonstrated when the average encoded transition density is considered in the next section.

5.3.5 Encoded Stream Transition Density

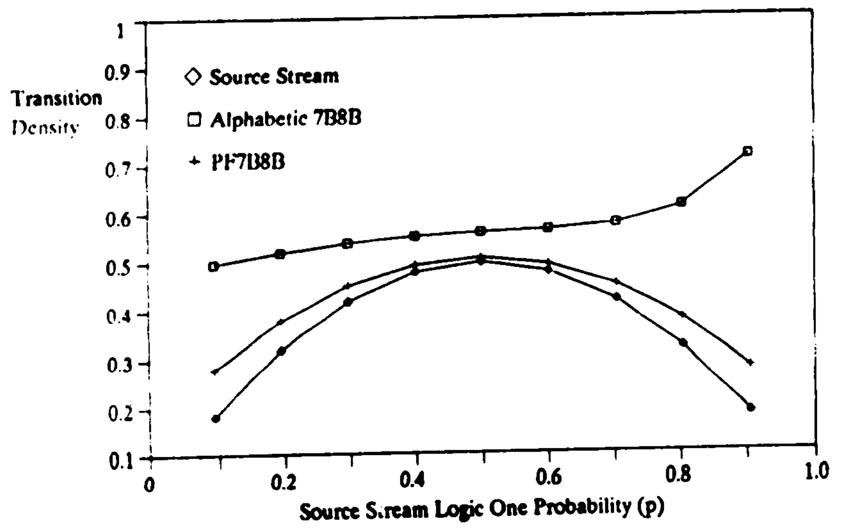
Simulation has shown that the lower bounds for Guided Scrambling encoded stream transition density given in Tables 5.1 through 5.6 are valid. Bounds on L_{max} for balanced Guided Scrambling configurations dictate that the lower bounds for encoded stream transition density given in Tables 5.2 through 5.5 hold. Simulation results which give the minimum number of transitions per encoded word when $d(x) = x^2+1$ demonstrate that values for td_{min} in Tables 5.1 and 5.6 are also valid.

But simulations have demonstrated that an encoded stream transition density much greater than the lower bound can be expected for a number of Guided Scrambling configurations and input sequence dc levels. Figure 5.13 plots, as a function of source stream logic one probability, source stream transition density and encoded stream transition density for the Alphabetic 7B8B, PF7B8B, and a host of Guided Scrambling codes. Simulation results for the following Guided Scrambling configurations are not given due to their redundant nature:

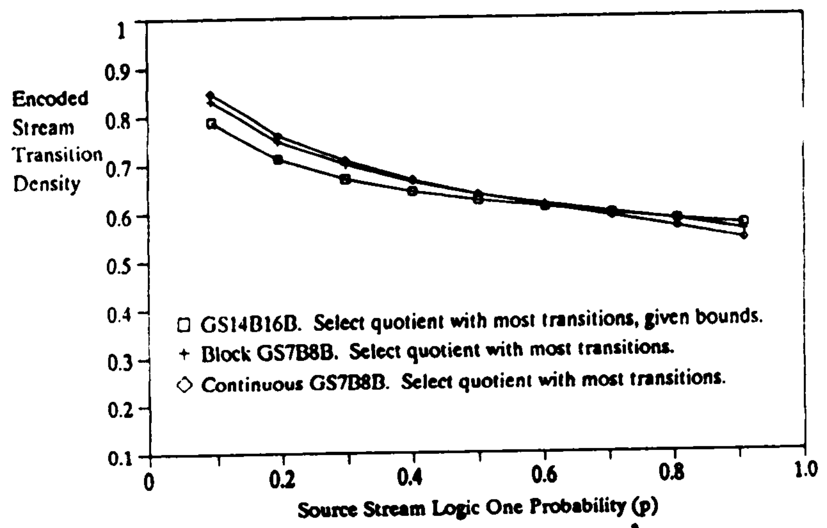
(1) The GS8B9B code demonstrated characteristics very similar to the GS7B8B code configured with $d(x) = x+1$. Consequently, its results are not reported.
(2) Variation of WRDS bounds for codes where their increase gives decreased L_{max} resulted in encoded stream transition densities which varied by less than 0.01 transitions/bit for any value of p. Accordingly, results are reported for only a single bounding condition.
(3) Since the dual augmenting bit Guided Scrambling codes simulated exhibited transition densities which varied by less than 0.02 transitions/bit for any one value of p, only simulation results for the GS14B16B code are shown.

The simulation results of this section clearly agree with the code spectra of Section 5.2 and the consecutive like bit distributions of Section 5.3.4. The Partially Flipped encoded stream transition density shows obvious symmetry about $p = 0.5$. The alphabetic code and single augmenting bit Block GS codes exhibit a high average encoded stream transition density when source dc is high. Codes configured with $d(x) = x^2+1$ and two augmenting bits transmit the most timing information when p is low. The statistics of continuously encoded Guided Scrambling bit sequences are nearly independent from source stream characteristics when the degree of the scrambling polynomial approaches that of the encoded word length. But note that these simulation results now clearly demonstrate that the performance of continuous Guided Scrambling with a scrambling polynomial of low degree retains strong dependence on source stream dc level. This source stream dc dependence is seen in the encoded stream transition density of the Continuous GS code configured with $n_a = 1$, $d(x) = x^2+1$. As mentioned in previous sections, this dependence is expected due to the low level of feedback generated through division register updating with this low degree scrambling polynomial.

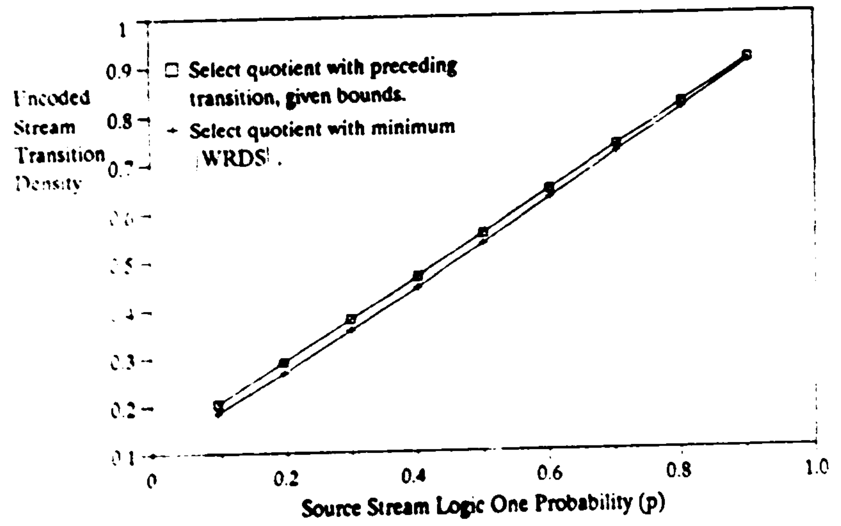
These results also provide the opportunity to introduce additional numerical merit figures to line code performance. When Guided Scrambling encoding is continuous and codewords are 8 bits in



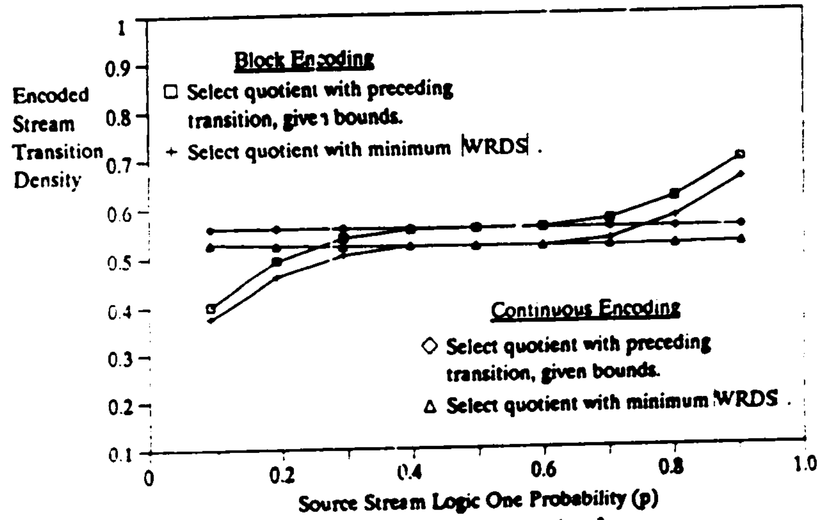
(a) Source and Conventionally Encoded Bit Streams



(b) Guided Scrambling, $d(x) = x^2 + 1$

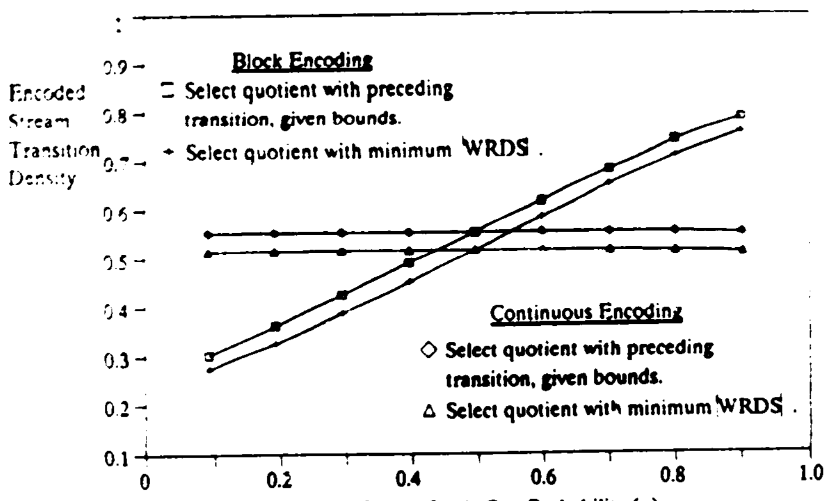


(c) GS7B8B, $d(x) = x + 1$

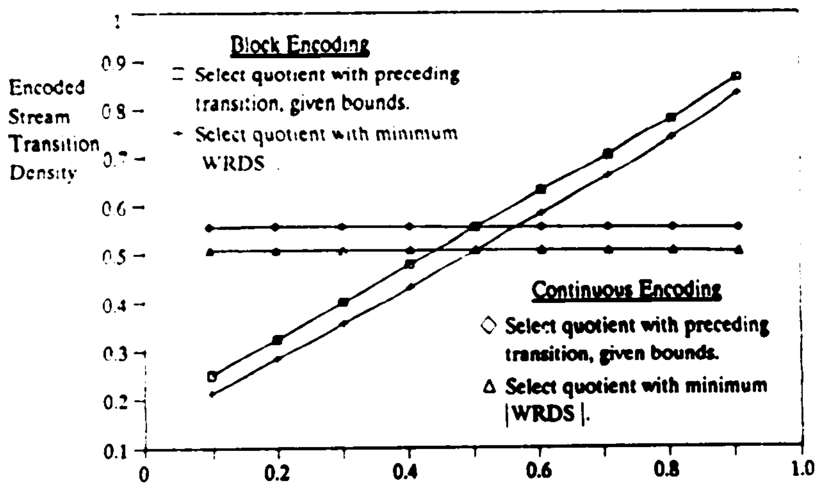


(d) GS7B8B, $d(x) = x^6 + x^4 + x^3 + x + 1$

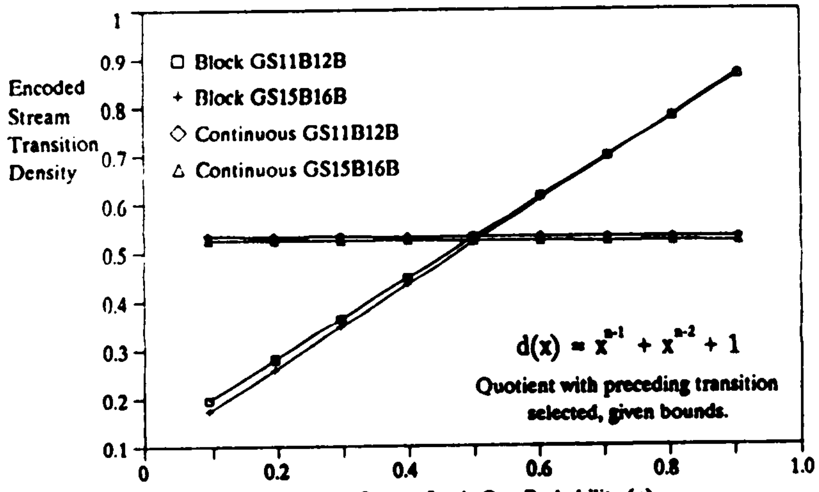
Figure 5.13: Encoded Stream Transition Density vs. Source Stream Logic One Probability



(e) GS7B8B, $d(x) = x^6 + x^5 + 1$



(f) GS7B8B, $d(x) = x^7 + x^6 + 1$



(g) GS Codes of higher efficiency

Enforced bounds correspond to maximum k in Tables 5.2 through 5.6.

Figure 5.13: Encoded Stream Transition Density vs. Source Stream Logic One Probability, continued.

length, selection of the quotient with a preceding transition, given enforcement of running digital sum bounds, consistently gives an encoded stream transition density of 0.56 transitions per bit. Inverting this value gives an average distance between transitions in the encoded bit sequence of 1.79 bits. These figures suggest performance which is superior to the Partially Flipped code for all source stream logic one probabilities, and indicate transmission of fewer level transitions than the alphabetic code rules only when source sequence logic one probability rises above 0.7.

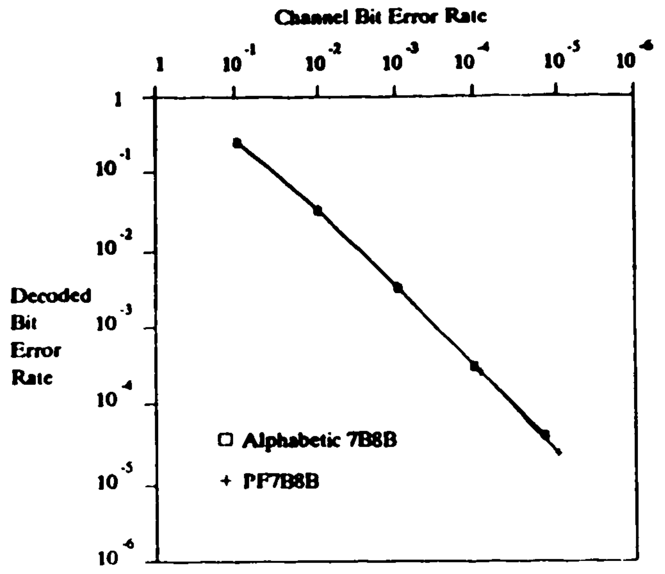
The transition density figures also clearly demonstrate that selection of the quotient with minimum |WRDS| results in the transmission of less timing information than configurations which select quotients based on word transitions. When $d(x) = x+1$, the impairment is approximately 0.02 transitions per bit. In other configurations, the degradation is consistently 0.05 transitions per bit.

Finally, the plots of Figure 5.13(g) show that Guided Scrambling performance with regard to average encoded stream transition density does not degrade significantly when code efficiency is increased. The number of transitions transmitted by the efficient block encoded configurations is lower than less efficient codes at low source stream logic one probabilities but does not decrease for high values of source dc. When encoding is continuous and the degree of scrambling polynomial is high, encoded stream transition density remains independent of source stream logic one probability. The 91.67% efficient code is characterized by a transition density of 0.54 transitions/bit. When efficiency is further increased to 93.75%, the transition density falls only to 0.53, a decrease of only 0.03 transitions/bit from the continuous GS7B8B codes.

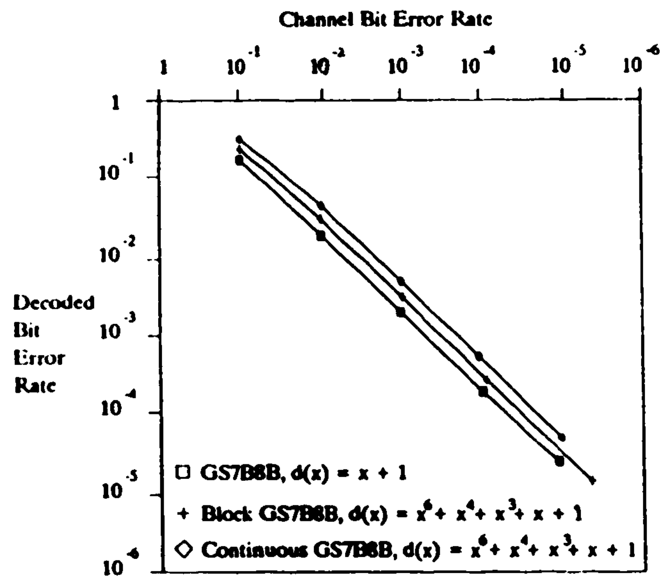
5.3.6 Error Extension

Simulation of the Guided Scrambling decoding rules has demonstrated that the proposed block and continuous decoding procedures will recover the original source sequence from encoded bit streams. Accurate source stream recovery is ensured when transmission is errorless. Error extension is exhibited when it is not so.

Figure 5.14 depicts simulated decoded error rates for the Alphabetic 7B8B and PF7B8B codes and selected Guided Scrambling configurations. Table 5.10 gives the average error extension these codes exhibited for received bit error rates less than or equal to 10^{-2}. Error extension simulation was restricted to relatively high bit error rates due to the limited sample size. Clearly, the alphabetic and Partially Flipped codes exhibit similar error extension. Guided Scrambling with a polynomial of low weight results in less error extension than the established line code techniques, but when a scrambling polynomial of high weight is used or continuous rather than block decoding is implemented, error extension increases.



(a) Conventional Line Codes



(b) Guided Scrambling

Figure 5.14: Simulated Error Extension

Code	Average Error Extension ($p = 0.5$, Channel BER$\leq 10^{-2}$)
Alphabetic 7B8B	3.31
PF7B8B	3.45
GS7B8B, $d(x) = x + 1$	1.97
Block GS7B8B, $d(x) = x^6 + x^4 + x^3 + x + 1$	3.21
Continuous GS7B8B, $d(x) = x^6 + x^4 + x^3 + x + 1$	5.01

Table 5.10: Average Simulated Error Extention

5.4 Performance Summary

The most significant points of the preceding discussion regarding Guided Scrambling line code performance are summarized as follows:

(1) Performance figures reported in Tables 5.1 through 5.7 make it clear that Guided Scrambling cannot enforce the tight constraints on encoded stream digital sum variation and maximum consecutive like bits that alphabetic codes provide. The best performance in low bit rate transmission systems still results from look-up table line code techniques. But in high bit rate systems where implementation of alphabetic codes is not yet possible, Guided Scrambling offers tighter bounds on L_{max} and DSV than its nearest rival, Partially Flipped coding.

(2) When each source word is augmented with two redundant bits and the scrambling polynomial is $d(x) = x^2+1$, Guided Scrambling results in the transmission of the greatest amount of timing information when source stream logic one probability is low. But it is also for low values of source dc that these Guided Scrambling configurations generate long like bit sequences with the greatest frequency. As source stream logic one probability increases, so too does the occurrence of two consecutive like bits in the encoded stream, and the frequency of occurrence of single bits and long like bit sequences decreases. Consequently, the majority of transmitted power shifts from occupying the frequency range near $fT = 0.5$ to $fT = 0.25$ and $fT = 0.75$.

(3) When GS encoding is performed on a word by word basis and augmentation is with a single bit per word, the encoded stream is characterized by the greatest number of level transitions when the source stream logic one probability is high. This increase in timing information is marked by an increase in the occurrence of stand-alone bits and a decrease in the occurrence of long like bit sequences. At high levels of source stream dc, most Block GS codes yield transmission of more level transitions than alphabetic encoding rules.

(4) Continuous Guided Scrambling with high degree scrambling polynomials results in transmission of a bit stream with long-term characteristics which do not depend upon the statistics of the source bit stream. The simplification in system design that this line code property offers is not available from any efficient conventional line coding technique. Continuous GS codes transmit more level transitions on average at low source dc levels than the Alphabetic 7B8B code and transmit more level transitions than Partially Flipped codes over the full range of source stream logic one probabilities.

(5) GS codes which select quotients on the basis of their associated transitions consistently transmit more level transitions than those which select the quotient exhibiting minimum |WRDS|.

(6) Guided Scrambling with low weight scrambling polynomials results in less error extension than conventional line code decoding procedures. When the degree of the scrambling polynomial is greater than the number of augmenting bits, GS decoding on a word by word basis results in less error extension than continuous decoding.

CHAPTER 6
CONCLUSION

This thesis has introduced Guided Scrambling, a new line coding technique developed specifically for high bit rate fiber systems. This chapter summarizes the development of this coding technique and offers suggestions for further work.

6.1 Thesis Summary

Following the introduction and discussion of notation in Chapter 1, Chapter 2 outlines characteristics required of line codes in fiber systems and discusses standard coding techniques which attempt to provide them. It is apparent from this discussion that little effort has been directed towards the development of efficient line codes which can be implemented at high rates. Most work has been concerned with easily implemented but inefficient line codes, codes which are inefficient and restricted to low rate systems due to look-up table implementation, or the probabilistic process of scrambling which generally results in better line code characteristics but cannot ensure in·provement.

Following a new interpretation of the encoding and decoding procedures involved in self-synchronizing scrambling, Chapter 3 introduces the concept of a new line coding technique which can be both highly efficient and easily implemented. To summarize, self-synchronizing scrambling can be viewed as continuous modulo-2 division of the source bit stream by a scrambling polynomial, and unscrambling as continuous multiplication. It is also observed that during scrambling division, the value of a single source dividend bit affects the value of many encoded quotient bits. This observation prompted inquiry into whether or not insertion of appropriately valued bits into the source bit stream prior to scrambling could guide the scrambling process to produce an encoded bit stream containing the properties of a good line code. Subsequent analysis and simulation has shown that such augmentation followed by scrambling with an appropriate polynomial, appropriate quotient selection and correct shift register updating will indeed generate an encoded bit stream with the required characteristics. The result is the new line coding technique called Guided Scr mbling.

The efficient implementation of Guided Scrambling is considered in Chapter 4. When the augmenting bits are inserted at regular intervals in the source bit stream, encoding can proceed through the creation of a number of augmented bit streams, each with different valued augmenting bits, followed by scrambling of each augmented stream with shift registers which implement modulo-2 division and the transmission of the quotient which best satisfies line code requirements. Decoding can be performed by unscrambling the received bit sequence and removing the augmenting bits. Derivation of appropriate scrambling polynomials and discussion of quotient selection mechanisms follows. It is also shown that, depending on how the encoder and decoder shift registers are updated, the transmitted bit stream will appear to be either one continuous quotient sequence or the concatenation of quotients of codeword length. The method by which the registers are updated and the corresponding format of the transmitted bit stream form the distinction between Continuous and Block Guided Scrambling.

Chapter 5 reports the performance of Guided Scrambling. Although it is found that GS codes cannot provide the tight constraints on encoded stream digital sum variation and maximum consecutive like bits enforced by look-up table codes, they do allow for tighter constraints than Partially Flipped codes. They also consistently result in the transmission of more level transitions than Partially Flipped codes and in some cases provide for the transmission of even more timing information than a well known alphabetic code. During discussion of Guided Scrambling performance it becomes clear that selection of code parameters will tailor code performance to meet individual requirements. Block GS codes transmit a large number of transitions at either high or low source stream dc levels, depending on the number of augmenting bits and the scrambling polynomial. Continuous GS codes with high degree scrambling polynomials provide encoded stream characteristics which are almost independent of source stream properties. This independence is not offered by conventional line code techniques.

Guided Scrambling then offers a new approach to line coding. It is simple in concept, based only on source stream augmentation prior to self-synchronizing scrambling and removal of the augmenting bits following unscrambling. It is simple to implement. Scrambling, unscrambling, and quotient selection can be implemented with combinational and sequential logic, bit stream augmentation and expurgation can be included in the multiplex and demultiplex operations which usually surround system transmission elements. It provides line code performance which is superior to conventional non-alphabetic coding techniques and approaches that of alphabetic codes. It must be considered a contender for implementation in high bit rate optical transmission systems.

6.2 Further Work

Since this thesis is the first exploration of this line coding technique, work remains regarding possible code configurations, performance characterization, and inclusion of other system features. The following are but a few avenues this further work might take.

(1) The derivation and performance characterization of other Guided Scrambling configurations could be undertaken. Code performance superior to that reported in this thesis might result when the following configuration possibilities are considered:

(a) other scrambling polynomials for single and dual augmenting bit GS codes. Only one polynomial has been proposed for codes using two augmenting bits, and the polynomials proposed for codes with single bit augmentation were only those which provided the required quotient set relationships with minimum weight. Other polynomials will also provide good line code performance. Of particular interest are high degree polynomials for use in Continuous Guided Scrambling configurations.

(b) augmentation with three or more augmenting bits per word.

(c) insertion of augmenting bits in positions other than the most significant of each word. Alternative placement is of most interest when augmentation is with multiple bits per word.

(d) variable rather than fixed augmenting bit placement.

(e) alternative methods of quotient selection.

(2) In this thesis, Guided Scrambling analysis and simulation has been restricted to the instance of random source data. When correlation is known to exist in the source bit stream, further analysis could be undertaken regarding optimum translation of this information into good line code characteristics through Guided Scrambling. Alternatively, if the random source stream is precoded and GS encoding is optimized for the statistics of this precoded stream, encoded stream characteristics superior to those reported here might result. This precoding could be performed through a preliminary attempt at Guided Scrambling or other easily implemented mechanisms.

(3) The possibility exists for a more extensive analysis of the relationship between the degree of the scrambling polynomial and the independence of continuously encoded bit stream characteristics from source stream logic one probability.

(4) Since the quotient selection mechanisms proposed in this work guarantee the selection of a single quotient from every selection set, it might be possible for Guided Scrambling decoders to automatically locate the position of augmenting bits in the decoded bit stream. The possibility and performance of GS decoder self-framing could be examined.

(5) The inclusion of features such as error detection and correction, system framing, and ancillary channels into Guided Scrambling could be considered due to the increase in code practicality their integration offers.

REFERENCES

[1] K. W. Cattermole, "Principles of digital line coding", *Int. J. Electronics*, vol 55, no. 1, pp. 3-33, July 1983.

[2] R. M. Brooks, A. Jessop, "Line coding for optical fiber systems", *Int J. Electronics*, vol 55, no. 1, pp. 81-120, July 1983.

[3] Y. .akasaki et al., "Optical Pulse Formats for Fiber Optic Digital Communications", *IEEE Transactions on Communications*, vol. COM-24, no. 4, pp. 404-413, April 1976.

[4] Y. Takasaki et al., "Two-level AMI line coding family for optical fibre systems", *Int. J. Electronics*, vol. 55, no. 1, pp. 121-131, July 1983.

[5] N. Yoshikai et al., "Line Code and Terminal Configuration for Very Large-Capacity Optical Transmission Systems", *IEEE Journal on Selected Areas in Communications*, vol. SAC-4, no. 9, pp. 1432-1437, Dec. 1986.

[6] H. Muller, "Bit Sequence Independence Through Scramblers in Digital Communication Systems", *Nachrichtentechn. Z.*, vol. 27, no. 12, pp. 475-479, 1974.

[7] J. E. Savage, "Some Simple Self-Synchronizing Digital Data Scramblers", *Bell System Technical Journal*, vol. 42, no. 2, pp. 449-487, Feb. 1967.

[8] R. Brooks, "7B8B Balanced Code with Simple Error Detecting Capability", *Electronics Letters*, vol.16, pp. 458-459, 1980.

[9] R. M. Brooks, "Design of a 7B8B Line Code with Improved Error Monitoring Capabilities", *Mathematical Topics in Telecommunications, Volume 2: Problems of Randomness in Communication Engineering*, John Wiley and Sons, New York, 1984, pp. 301-318.

[10] A. J. Sharland, A. Stevenson, "A simple in-service error detection scheme based on the statistical properties of line codes for optical fibre systems". *Int. J. Electronics*, vol. 55, no. 1, pp. 141-158, July 1983.

[11] R. Petrovic, "5B6B Optical Fibre Line Code Bearing Auxiliary Signals", *Electronics Letters*, vol. 24, no. 5, pp. 274-275, March 1988.

[12] W. D. Grover, E. A. Munter, "Transmitting a Digital Signal and an Additional Signal", Canadian Patent no. 1209706, August 12 1986.

[13] P. Radev, G. Stoyanov, "A New 1B2B Line Code for Digital Fibre Optic Transmission Systems", *Electronics Letters*, vol. 20, no. 20, pp. 355-356, 1984.

[14] J. M. Griffiths, "Binary Code Suitable for Line Transmission", *Eletronics Letters*, vol. 5, no. 4, Feb. 1969, pp. 79-81.

[15] O. N. Porokho, P. Radev, "3B2T-RBS - An Efficient Transmission Method for Digital Fibre-Optic Communications", *Electronics Letters*, vol. 22, no. 21, pp. 1125-1126, 1986.

[16] R. O. Carter, "Low-Disparity Binary Coding System", *Electronics Letters*, vo. 1, no. 3, pp. 67-68, May 1965.

[17] W. A. Krzymien, "A New Class of Balanced Line Codes for Optical Fibre Transmission Systems", *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria BC, pp. 316-318, June 4-5 1987.

[18] T. V. Muoi, "Receiver Design for High-Speed Optical-Fiber Systems", *Journal of Lightwave Technology*, vol. LT-2, no. 3, pp. 243-267, June 1984.

[19] S. D. Personick, "Receiver Design for Optical Fiber Systems", *Proceedings of the IEEE*, vol. 65, no. 12, pp. 1670-1678, Dec. 1977.

[20] "Vocabulary of pulse code modulation (PCM) and digital transmission terms", *CCITT Recommendation G.702*.

[21] C. J. Byrne et al., "Systematic Jitter in a Chain of Digital Regenerators", *Bell System Technical Journal*, vol. 42, no. 6, pp. 2679-2714, Nov. 1963.

[22] W. W. Peterson, E. J. Weldon Jr., *Error-Correcting Codes*, 2nd ed., The MIT Press, Cambridge Massachusettes, 1971.

[23] G. L. Cariolaro et al., "Analysis of codes and spectra calculations", *Int. J. Electronics*, vol. 55, no. 1, pp. 35-79, 1983.

[24] G. L. Cariolaro, G. P. Tronca, "Spectra of Block Coded Digital Signals", *IEEE Transactions on Communications*, vol. COM-22, no. 10, pp. 1555-1564, Oct. 1974.

[25] D. A. Huffman, "The Synthesis of Linear Sequential Coding Networks", *Proceedings of the Third London Symposium on Information Theory*, London England, pp. 77-95, Sept. 12-16 1955.

[26] R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, Reading Massachusettes, 1983.

[27] J. G. Proakis, *Digital Communications*, McGraw-Hill, USA, 1983.

APPENDIX 1

NOTES ON GUIDED SCRAMBLING

This appendix contains several observations on the process of Guided Scrambling. The relationship between augmented source words, quotients, and remainders in GS encoding is first established. For single bit augmentation, the relationship between the number of transitions in r(x) and the number of transitions in the quotients which comprise the selection set is considered, followed by a derivation of the restrictions on r(x) which ensure transmission of a balanced sequence.

A1.1 Relationship of augmented source words, quotients, and remainders in GS encoding

A1.1.1 Block GS codes

To establish the relationship amongst Guided Scrambling bit sequences, consider once again the encoding process:

$$\begin{aligned} q(x) &= Q_{d(x)}[a(x)\,x^d] \\ rm(x) &= R_{d(x)}[a(x)\,x^d]\ , \end{aligned} \tag{A1.1}$$

where:

Q	=	evaluation of the quotient generated through division of its argument by its subscripted polynomial,
R	=	similar evaluation of a remainder,
a(x)	=	source word augmented with either a zero or a one, length n,
d(x)	=	scrambling polynomial, of degree d,
q(x)	=	n-bit quotient,
rm(x)	=	d-bit remainder,

and arithmetic is modulo-2. These equations can be combined in the more conventional form:

$$a(x)\,x^d = q(x)\,d(x) + rm(x). \tag{A1.2}$$

GS decoding is performed by first multiplying the received sequence by d(x) to form a product p(x). In the absence of errors:

$$\begin{aligned} p(x) &= q(x)\,d(x) \\ &= a(x)\,x^d + rm(x)\ . \end{aligned} \tag{A1.3}$$

Since the polynomial $[a(x)\,x^d]$ contains zeros in the d least significant bit positions and the remainder rm(x) is of at most degree (d-1), coefficients of the p(x), a(x) and rm(x) sequences in (A1.3) are related by:

$$\begin{aligned} c_i^{p(x)} &= c_i^{rm(x)}\ , & 0 \le i \le (d-1) \\ c_{i+d}^{p(x)} &= c_i^{a(x)}\ , & 0 \le i \le (n-1)\ , \end{aligned} \tag{A1.4}$$

where superscripts and subscripts denote the sequence to which the coefficients belong and their degree respectively.

Long division expansion of (A1.1) reveals that the coefficients of the quotient sequence are related to those of the augmented sequence and scrambling polynomial by:

$$c_{(n-1-i)}^{q(x)} = c_{n-1-i}^{a(x)} + \begin{cases} \sum_{j=0}^{i-1} c_{n-i+j}^{q(x)} c_{d-j+1}^{d(x)} , & 0 \le i \le (d-1) \\ \sum_{j=0}^{d-1} c_{n-i+j}^{q(x)} c_{d-j+1}^{d(x)} , & d \le i \le (n-1) . \end{cases} \qquad \text{(A1.5)}$$

Since this recursive relationship demonstrates that quotient coefficients depend only on coefficients of a(x), d(x), and previously evaluated coefficients of the same quotient, it is clear that division by d(x) maps each a(x) to a particular q(x).

Expansion of (A1.3) into shift and add operations and use of the relationship between coefficients of a(x) and p(x) given in (A1.4) reveals that:

$$c_{n-1-i}^{a(x)} = \sum_{j=0}^{i} c_{n-1-j}^{q(x)} c_{d-i+j}^{d(x)} , \qquad 0 \le i \le (n-1) . \qquad \text{(A1.6)}$$

Implicit in (A1.6) is the recovery of a different a(x) sequence from each q(x). Equations (A1.5) and (A1.6) then define a one-to-one mapping between a(x) and q(x).

An important result of this one-to-one mapping is that when no restrictions are placed on the form of the source bit stream and augmentation is with all combinations of augmenting bits, all 2^n quotient sequences can appear in a quotient selection set. Performance analysis will proceed given the possibility of any quotient in the selection set.

Since the remainders consist of d bits, it is obvious that when $d \neq n$ the mapping between a(x) and rm(x) cannot be one-to-one. When $d \le n$, the expansion used to derive (A1.6) also reveals that:

$$c_i^{rm(x)} = \sum_{j=0}^{i} c_j^{q(x)} c_{i-j}^{d(x)} , \qquad 0 \le i \le (d-1) . \qquad \text{(A1.7)}$$

This equation demonstrates that each combination of the d least significant bits of q(x) maps to a single rm(x) through multiplication by d(x). Since each combination of the d least significant bits of q(x) can occur with each combination of its (n-d) most significant bits, and the mapping between q(x) and a(x) is one-to-one, then when $d < n$, each remainder results from the division of 2^{n-d} different a(x) sequences. When $d = n$, each a(x) will generate a unique remainder through division by d(x). When $d > n$, only a subset of the 2^d possible remainders will be generated.

A1.1.2 Continuous GS Codes

When encoding division registers are continuously updated rather than cleared, the mapping from a(x) to q(x) is not one-to-one. The contents of the division registers prior to division also play a role in the generation of q(x).

Let the superscript t denote a sequence presently being encoded and the superscript (t-1) denote a sequence involved in the encoding of the previous word. Let the division register be clear prior to division (i.e. $rm^{t-1}(x) = 0$) and introduce $a^t(x)\,x^d$. The quotient generated through n shifts of the register is:

$$q^t(x) \;=\; Q_{d(x)}[a^t(x)\,x^d]\ . \tag{A1.8}$$

Now, let the register contain a non-zero $rm^{t-1}(x)$ but let $a^t(x)$ be the all zero sequence. n shifts of the division register produce the quotient:

$$q^t(x) \;=\; Q_{d(x)}[rm^{t-1}(x)\,x^n]\ . \tag{A1.9}$$

When both $a^t(x)$ and $rm^{t-1}(x)$ are non-zero, (A1.8) and (A1.9) combine to give:

$$\begin{aligned} q^t(x) &= Q_{d(x)}[a^t(x)\,x^d] + Q_{d(x)}[rm^{t-1}(x)\,x^n] \\ &= Q_{d(x)}[(a^t(x) + rm^{t-1}(x)\,x^{n-d})\,x^d] \\ &= Q_{d(x)}[a^{t\prime}(x)\,x^d]\,, \end{aligned} \tag{A1.10}$$

where:

$$a^{t\prime}(x) \;=\; a^t(x) + rm^{t-1}(x)\,x^{n-d}\ . \tag{A1.11}$$

Following the reasoning applied to Block GS codes, it is clear that the mapping from $a'(x)$ to q(x) is one-to-one and that when $d \le n$, each rm(x) is generated from 2^{n-d} $a'(x)$ sequences. Once again, all n-bit quotients might appear in the quotient selection set when no restrictions are placed on the source bit stream and source words are augmented with all possible combinations of augmenting bits.

A1.2 Relationship of transitions in r(x) to transitions in $q_0(x)$ and $q_1(x)$, $n_a = 1$

To determine the relationship between the number of transitions in r(x) and the number of transitions in the two-word quotient selection set generated through single bit source word augmentation, consider first the modulo-2 addition of two bit sequences. If the addition of these sequences is viewed in terms of transition addition rather than bit addition, it is noted that transitions also undergo modulo-2 addition. The rules for this addition can be explicitly stated as:

no transition +	no transition	=	no transition	(A1.12)
no transition +	transition	=	transition	
transition +	no transition	=	transition	
transition +	transition	=	no transition .	

For example, consider the modulo-2 addition of the following sequences:

$$\begin{array}{rr} & 00010 \\ + & 00110 \\ \hline & 00100\ . \end{array}$$

In terms of transition operations, the addition of these sequences can be viewed as:

$$\begin{array}{rc} & - - x\,x \\ + & - x - x \\ \hline & - x\,x - \end{array} \qquad \begin{array}{l} x = \text{transition} \\ - = \text{no transition .} \end{array}$$

That the transitions undergo addition according to (A1.12) is obvious.

Consider now the number of transitions in the sequences $q_0(x)$ and $q_1(x)$ given the fixed sequence r(x) through which they are related. From the rules given for transition addition it is easy to see that:

$$t_{1,int} = t_{0,int} + t_{r,int} - 2t_s \,, \tag{A1.13}$$

where:

$t_{0,int}$ = number of transitions within $q_0(x)$
$t_{1,int}$ = number of transitions within $q_1(x)$
$t_{r,int}$ = number of transitions within r(x)
t_s = number of transitions in $q_0(x)$ and r(x) which occupy the same transition opportunities.

Solutions to this equation exist at integer values within the shaded rectangle in Figure A1.1.

Let t_{int} be the number of transitions within the quotient selected by the encoder decision device. If the quotient with the greatest number of transitions is selected, it is clear from Figure A1.1 that:

$$t_{int} \geq \begin{cases} \dfrac{t_{r,int}}{2}\,, & t_{r,int} = \text{even} \\[2ex] \dfrac{t_{r,int}+1}{2}\,, & t_{r,int} = \text{odd}\,. \end{cases} \tag{A1.14}$$

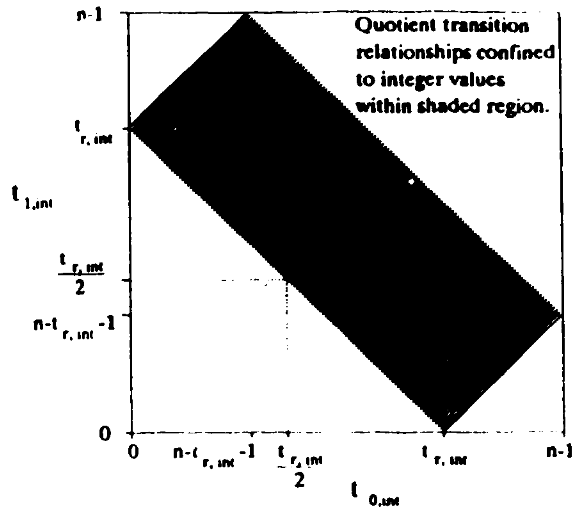


Figure A1.1: Relationships between transitions in $q_0(x)$ and $q_1(x)$

A1.3 Restrictions on r(x) for balanced GS encoding, $n_a = 1$

Balanced transmission implies that the running digital sum of the transmitted sequence is bounded, which also means that bounds exist on the transmitted stream word-end running digital sum. The requirement for enforcement of WRDS bounds places restrictions on the form of r(x). The derivation of these restrictions, when augmentation is with a single bit per word, follows.

Since the word disparity of an odd length sequence is always odd and the WD of an even length sequence is always even, enforcement of WRDS bounds, given the encoder selection between two quotient sequences, requires that the following condition be true:

$$\left.\begin{array}{l}\text{one sequence has WD} > 0\\ \text{one sequence has WD} < 0\end{array}\right\}\quad n \text{ odd },$$
$$\left.\begin{array}{l}\text{one sequence has WD} \geq 0\\ \text{one sequence has WD} \leq 0\end{array}\right\}\quad n \text{ even }. \tag{A1.15}$$

Consider selection between $q_0(x)$ and $q_1(x)$. Let #ones and #zeros denote the number of marks and number of spaces respectively in a bit sequence. Since n-bit sequences consist of either marks or spaces:

$$\#\text{ones} \;=\; n - \#\text{zeros} \;. \tag{A1.16}$$

When the subscripts 0 and 1 denote attributes of $q_0(x)$ and $q_1(x)$ respectively, Equation (A1.15) can be restated in the following manner:

$$\left.\begin{array}{lcll}\#\text{ones}_0 & + & \#\text{ones}_1 & = n\\ \#\text{zeros}_0 & + & \#\text{zeros}_1 & = n\end{array}\right\}\quad n \text{ odd },$$
$$\left.\begin{array}{lcll}\#\text{ones}_0 & + & \#\text{ones}_1 & \leq (n+1)\\ \#\text{zeros}_0 & + & \#\text{zeros}_1 & \leq (n+1)\end{array}\right\}\quad n \text{ even }. \tag{A1.17}$$

Consider now the formation of $q_1(x)$ from $q_0(x)$ through addition with r(x). Let n_{nc} denote the number of $q_0(x)$ bits not complemented during addition with r(x). If $n_{nc} = 0$, then:

$$\#\text{ones}_1 \;=\; \#\text{zeros}_0 \;.$$

If $n_{nc} = 1$, then:

$$\begin{array}{lcll}\#\text{ones}_1 & = & \#\text{zeros}_0 + 1 & \text{if uncomplemented bit} = 1\\ \#\text{ones}_1 & = & \#\text{zeros}_0 - 1 & \text{if uncomplemented bit} = 0 \;,\end{array}$$

and so on. These relationships for $0 \leq n_{nc} \leq 4$ are given in Table A1.1. Noting the progression of this table, it is clear that:

$$\begin{array}{lcll}\#\text{ones}_1 & = & \#\text{zeros}_0 & \pm\, i\\ \#\text{zeros}_1 & = & \#\text{ones}_0 & \mp\, i\end{array}\qquad i = \begin{cases}1, 3, 5 \ldots n_{nc}, & n_{nc} \text{ odd}\\ 0, 2, 4 \ldots n_{nc}, & n_{nc} \text{ even}\;.\end{cases} \tag{A1.18}$$

n_{uc}	uncomplemented bit values	$\#ones_1 = \#zeros_0 +$
0	-	0
1	0 1	- 1 + 1

2	0 0
	0 1
	1 1
		- 2
		0
		+ 2

3	0 0 0
	0 0 1
	0 1 1
	1 1 1
		- 3
		- 1
		+ 1
		+ 3

4	0 0 0 0
	0 0 0 1
	0 0 1 1
	0 1 1 1
	1 1 1 1
		- 4
		- 2
		0
		+ 2
		+ 4

Table A1.1: Relationship between #ones and [illegible]os in $q_0'(x)$ and $q_1(x)$

Combining (A1.16) and (A1.18) yields:

$$\begin{aligned} \#ones_0 + \#ones_1 &= n \pm i \\ \#zeros_0 + \#zeros_1 &= n \mp i \end{aligned} \quad i = \begin{cases} 1, 3, 5 \ldots n_{uc}, & n_{uc} \text{ odd} \\ 0, 2, 4 \ldots n_{uc}, & n_{uc} \text{ even} . \end{cases} \tag{A1.19}$$

If the restrictions of Equation (A1.17) are placed on the value of i in (A1.19), it is immediately evident that:

$$i = \begin{cases} 0, & n \text{ odd} \\ 0 \text{ or } 1, & n \text{ even} . \end{cases} \tag{A1.20}$$

From the definition of i first given in Equation (A1.18), it is clear that its value is restricted by the value of n_{uc}. Rewriting the condition of Equation (A1.20):

To ensure that the quotient selection set will always contain a quotient which allows for transmission of a balanced bit stream when augmentation is with a single bit per word, the number of bits in $q_0(x)$ not complemented to form $q_1(x)$ must be:

$$n_{uc} = \begin{cases} 0, & n = \text{odd} \\ 0 \text{ or } 1, & n = \text{even} . \end{cases} \tag{A1.21}$$

APPENDIX 2

,UIDED SCRAMBLING PERFORMANCE ANALYSIS

This appendix presents analysis concerning the performance of the following Guided Scrambling configurations:

(1) $n_a = 1,\ d(x) = x^2+1$

(2) $n_a = 1,\ d(x) = x+1$

(3) $n_a = 1,\ d(x) \in D_{mods}(x)$

(4) $n_a = 1,\ d(x) = x^{n-2}+x^{n-3}+1$

(5) $n_a = 1,\ d(x) = x^{n-1}+x^{n-2}+1$

(6) $n_a = 2,\ d(x) = x^2+1$.

Areas of performance which are analyzed include bounds on encoded stream WRDS and DSV, maximum encoded stream consecutive like bits, minimum encoded stream transition density, and decoder error extension. The effect of various quotient selection mechanisms upon GS performance is considered. A tabular summary of performance for the best selection mechanism is given for each configuration. To avoid redundant derivation, analysis will build upon that which precedes it as mucl as possible.

A2.1 $n_a = 1, d(x) = x^2 + 1$

As shown in Chapter 4, the use of $d(x) = x^2+1$ when augmentation is with a single bit per word cannot provide balanced transmission. Guidec ambling with this polynomial should then seek to maximize the number of level transitions transmitted.

A discussion of several quotient selection mechanisms for this configuration is followed by performance analysis and determination of an optimal selection device. Fundamental analysis procedures and equations derived here will be used during analysis of other GS configurations.

A2.1.1 Quotient Selection Alternatives

There are several selection mechanisms which can be used to select the quotient which contains the greatest number of transitions. Only devices which ensure selection of a single quotient in every situation are considered here. Before considering the mechanism alternatives, note that:

(1) the sequence which relates the quotients through modulo-2 addition is an alternating sequence,

(2) due to this relationship, where one quotient has a transition, the other does not,

(3) an odd length sequence contains an equal number of transitions, an even length sequence an odd number of transitions, and

(4) an even number of transitions can be equally divided between the two quotients.

In light of these observations, the following encoder decision mechanisms are proposed:

n = odd:

(1) Select the quotient with the greatest number of transitions, where the transition count is confined to the transition opportunities which occur within the word. In the event of a tie in the number of transitions, select the quotient which contains a transition at a predetermined to_i, $(n-1) \geq i \geq 0$. The optimal value for i must be determined.

(2) Select the quotient with the greatest number of transitions, where the transition count includes the first transition opportunity $to_{(n-1)}$. The quotient alternatives will never contain an equal number of transitions.

n = even:

(1) Select the quotient with the greatest number of transitions, where the transition count does not include the first opportunity. The quotient alternatives will never contain an equal number of transitions.

(2) Select the quotient with the greatest number of transitions, where the transition count includes the first opportunity. In the event of an equal number of transitions in the quotients, select the quotient with a transition at a predetermined to_i, $(n-1) \geq i \geq 0$.

These decision alternatives will be considered in the following performance analysis.

A2.1.2 Performance Analysis

Due to the unbalanced transmission which results from this GS configuration, only the minimum encoded stream transition density, maximum consecutive like bits, and error extension can be evaluated.

A2.1.2.1 Minimum encoded stream transition density

As found in Section 4.2.2, when the quotient selection set consists of a pair of quotients that are related by the alternating sequence, one quotient will always contain at least $t_{int,min}$ transitions, where:

$$t_{int,min} = \begin{cases} \frac{n-1}{2}, & n \text{ odd} \\ \frac{n}{2}, & n \text{ even}. \end{cases}$$

Consider the situation when n is odd and the decision mechanism includes $to_{(n-1)}$ in the transition count. Given that the most significant bits of the quotient sequences differ in value, a quotient which contains $t_{int,min}$ transitions and a preceding transition can always be selected. In this case, the minimum number of transitions associated with each transmitted word becomes $^{(n+1)}/_{2}$.

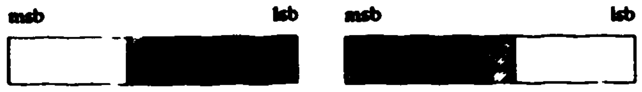
n	Transition Count	td_{min}
odd	Excludes $to_{(n-1)}$	$\frac{1}{2} - \frac{1}{2n}$
	Includes $to_{(n-1)}$	$\frac{1}{2} + \frac{1}{2n}$
even	Includes or Excludes $to_{(n-1)}$	$\frac{1}{2}$

Table A2.1: Minimum Encoded Stream Transition Density.
$n_a = 1, d(x) = x^2+1$

Table A2.1 gives the minimum encoded stream transition density for each mechanism being considered where td_{min} is found through the division of the minimum number of transitions associated with each quotient by the quotient length.

A2.1.2.2 Maximum consecutive like bits

Given restrictions on the minimum number of transitions in each se[illegible] quotient, it is clear that a quotient composed entirely of bits of one value would never be selected. A string of consecutive like bits can therefore span portions of at most two words, as shown in Figure A2.1. To determine the maximum number of consecutive like bits, the maximum number of like bits which can occur at the start and end of a word must be established. Denote consecutive like bits at the end of a word a tail, and the [illegible] num[illegible] [illegible] [illegible]r Word(A) in Figure A2.1 as $T_{A,max}$. Similarly, denote consecutive like bits at the start of a word a head, and their maximum in Word(B) as $H_{B,max}$.



Word(A) Word(B)

▨ = consecutive like bits

Figure A2.1: Situation for maximum consecutive like bits.
$n_a = 1, d(x) = x^2+1$

appear in the encoded bit stream.

Derivation of $T_{A,max}$ and $H_{B,max}$ will proceed in two stages:

(1) evaluation of how many bits have the potential to appear consecutively at the head or tail end of a word, and

(2) determination of whether words which exhibit these heads or tails would be selected in the appropriate situation.

To proceed with the first stage of the derivation, note that:

$$\begin{aligned} \#\text{ones} - \#\text{zeros} &= WD \\ \#\text{ones} - (n - \#\text{ones}) &= WD \\ \#\text{ones} &= \frac{n + WD}{2} . \end{aligned} \quad \text{(A2.1a)}$$

Similarly,

$$\#\text{zeros} = \frac{n - WD}{2} . \quad \text{(A2.1b)}$$

Now, let the term *majority bits* refer to the bits whose polarity has occurred most frequently within a word and the term *minority bits* refer to those whose value has occurred less frequently. Let a *stand-alone* bit be one which is found between bits of opposing polarity. Let the minimum number of transitions required within each quotient be denoted $t_{int,req}$. If $t_{int,req}$ transitions are satisfied solely with stand-alone bits, then:

$$\#\text{stand-alone bits} = \text{Rnd}\left(\frac{t_{int,req}}{2}\right) ,$$

where the Rnd operator indicates rounding to the next nearest integer. If the minority bits are all stand-alone bits, then:

$$\begin{aligned} |WD|_{max} &= (n - \#\text{minority bits}) - \#\ \text{minority bits} \\ &= n - 2\,\text{Rnd}\left(\frac{t_{int,req}}{2}\right) . \end{aligned} \quad \text{(A2.2)}$$

It is logical to expect that a maximum length head or tail could appear within a word which contains the greatest possible number of majority bits. The number of majority bits is then limited by the requirement for $t_{int,req}$ transitions within the word. If the majority bits which are not part of the head or tail stand alone, satisfaction of the requirement for $t_{int,req}$ transitions is with the fewest possible number of these majority bits, leaving a maximum length head or tail. Since:

$$\text{number of transitions required within a word} = \text{one transition from end of head or start of tail} + 2\begin{bmatrix}\text{number of stand-alone}\\ \text{majority bits}\end{bmatrix} - \begin{cases}1, & \text{if one of the stand alone majority bits is at an end of the word}\\ 0, & \text{if all stand-alone majority bits are embedded within minority bits}\end{cases},$$

a value for maximum head or tail length can be established from:

$$t_{int,req} = 1 + 2[\text{\# majority bits} - (H_{max}, T_{max})] - \begin{cases}1\\0\end{cases}, \qquad \text{(A2.3)}$$

where the value of the last constant can be determined from the even or odd nature of $t_{int,req}$. Combining (A2.1a) with (A2.3) yields:

$$t_{int,req} = 1 + 2[(n + WD) - (H_{max}, T_{max}|_{ones})] - \begin{cases}1\\0\end{cases}$$

$$H_{max}, T_{max}|_{ones} = \text{Rnd}\left[\frac{(n + WD - t_{int,req})}{2}\right]. \qquad \text{(A2.4a)}$$

Similarly:

$$H_{max}, T_{max}|_{zeros} = \text{Rnd}\left[\frac{(n - WD - t_{int,req})}{2}\right]. \qquad \text{(A2.4b)}$$

Maximum head and tail lengths can be calculated from Equations (A2.4) using the maximum values for word disparity from Equation (A2.2) and the appropriate value of $t_{int,req}$. The following reasoning is used to establish the value of $t_{int,req}$ for calculation of $T_{A,max}$ given $n_a = 1$, $d(x) = x^2 + 1$ and odd n:

(1) If the transition count does not included $to_{(n-1)}$, ${}^{(n-1)}/_2$ transitions must occur within each selected quotient. However, if one quotient contains ${}^{(n-1)}/_2$ transitions, so does the other. If this tie is broken by selecting the word with a transition within the first ${}^{(n-1)}/_2$ transition opportunities, a quotient with the maximum length tail given $t_{int,req} = {}^{(n-1)}/_2$ can be selected. But if the word with a transition within the last ${}^{(n-1)}/_2$ transition opportunities is selected, a word which contains a shorter tail than the maximum will always be selected. The largest tail possible in this situation will be one within a word which is selected when there is no tie. This word must contain a minimum of ${}^{(n+1)}/_2$ transitions.

(2) If the transition count includes $to_{(n-1)}$, the overall transition count must be greater than or equal to ${}^{(n+1)}/_2$. Since one of these transitions may occur in the first transition opportunity, the value of $t_{int,req}$ which must be satisfied is ${}^{(n-1)}/_2$.

Similar reasoning can be used to establish the value of $t_{tail,req}$ used when n is even. The results of this reasoning and the corresponding values of $T_{A,max}$ are given in Table A2.2.

It remains to be shown that words which exhibit these tails would be chosen when they are present in the quotient selection set. Tables A2.3 (a)-(d) list quotients which have these tail lengths and would be selected in preference to their alternative. By noting the pattern of the words in the tables, it is clear that words with tails equal to the values given in Table A2.2 would be selected regardless of quotient length.

The maximum head length possible in words without preceding transitions can be established through similar reasoning and calculation. Table A2.4 gives values for $t_{tail,req}$ and $H_{B,max}$ for the various quotient selection mechanisms. Tables A2.5 (a)-(d) give quotients which exhibit these heads and would be selected if they were present in the quotient selection set. The pattern of these tables indicate that such words exist for any n.

And now, after all that work, by combining the values of $T_{A,max}$ and $H_{B,max}$ in Tables A2.2 and A2.4 it is found that regardless of the encoder decision mechanism, when $n_a = 1$ and $d(x) = x^2+1$:

$$L_{max} = n \ .$$

A2.1.2.3 Error Extension

As stated in Section 2.3.1, self-synchronizing scrambling is susceptible to error extension during decoding. In that form of scrambling, an upper bound for the multiplication of transmission errors was given in Equation (2.2) as:

$$\text{\# of decoded errors} \leq (\text{\# of transmission errors}) * (\text{weight of scrambling polynomial}) \ .$$

More exact expressions for error extension can be derived for this Guided Scrambling configuration for both continuous and block decoding.

Continuous Decoding:

Consider the continuous multiplication of a received bit stream by $d(x) = x^2+1$, followed by removal of every nth bit. Since each transmission error is multiplied by the weight-2 scrambling polynomial, a single transmission error will map to two decoded errors. However, if two errors spaced one bit apart occur during transmission, the modulo-2 addition of the extended error patterns results in only two decoded errors rather than four. When the probability of an error during transmission is denoted $Pe_{channel}$, this pattern of transmission errors will occur with the probability:

$$Pe_{channel}^{\ 2}(1-Pe_{channel}) \ .$$

If three errors occur during transmission, all spaced one bit apart, again only two errors will be decoded instead of the six expected. The probability of this error pattern occurring is:

$$Pe_{channel}^{\ 3}(1-Pe_{channel})^2 \ .$$

n	Selection Mechanism	$t_{int,min}$	$T_{A,max}$
odd	Exclude $to_{(n-1)}$ from transition count. If counts equal, select the quotient with a transition at to_i, $(n-1) \ge i \ge (\frac{n+1}{2})$	$\frac{n-1}{2}$	$\frac{n+1}{2}$
	Exclude $to_{(n-1)}$ from transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n-1}{2}) \ge i \ge 0$	$\frac{n+1}{2}$	$\frac{n-1}{2}$
	Include $to_{(n-1)}$ in transition count.	$\frac{n-1}{2}$	$\frac{n+1}{2}$
even	Exclude $to_{(n-1)}$ from transition count.	$\frac{n}{2}$	$\frac{n}{2}$
	Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(n-1) \ge i \ge (\frac{n}{2}-1)$	$\frac{n}{2} - 1$	$\frac{n}{2} + 1$
	Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n}{2} - 2) \ge i \ge 0$	$\frac{n}{2}$	$\frac{n}{2}$

Table A2.2: Maximum Quotient Tail Lengths.
$n_a = 1, d(x) = x^2+1$

n	Quotient	Tail
5	10111	3
7	0101111	4
9	101011111	5
11	01010111111	6

(a) n = odd, $to_{(n-1)}$ included, or $to_{(n-1)}$ excluded and secondary selection = to_i,

$$(n-1) \geq i \geq \left(\frac{n+1}{2}\right)$$

n	Quotient	Tail
5	01011	2
7	1010111	3
9	010101111	4
11	10101011111	5

(b) n = odd, $to_{(n-1)}$ excluded, with secondary selection = to_i,

$$\left(\frac{n-1}{2}\right) \geq i \geq 0$$

n	Quotient	Tail
6	010111	3
8	10101111	4
10	0101011111	5
12	101010111111	6

(c) n = even, $to_{(n-1)}$ excluded, or $to_{(n-1)}$ included and secondary selection = to_i,

$$\left(\frac{n}{2}-2\right) \geq i \geq 0$$

n	Quotient	Tail
6	101111	4
8	01011111	5
10	1010111111	6
12	010101111111	7

(d) n = even, $to_{(n-1)}$ included, with secondary selection = to_i,

$$(n-1) \geq i \geq \left(\frac{n}{2}-1\right)$$

Table A2.3: Quotients which exhibit $T_{A,max}$
$n_a = 1$, $d(x) = x^2+1$

n	Selection Mechanism	$t_{int,min}$	$H_{n,max}$
odd	Exclude $to_{(n-1)}$ from transition count. If counts equal, select the quotient with a transition at to_i, $(n-1) \geq i \geq (\frac{n+1}{2})$	$\frac{n+1}{2}$	$\frac{n-1}{2}$
	Exlude $to_{(n-1)}$ from transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n-1}{2}) \geq i \geq 0$	$\frac{n-1}{2}$	$\frac{n+1}{2}$
	Include $to_{(n-1)}$ in transition count.	$\frac{n+1}{2}$	$\frac{n-1}{2}$
even	Exclude $to_{(n-1)}$ from transition count.	$\frac{n}{2}$	$\frac{n}{2}$
	Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(n-1) \geq i \geq (\frac{n}{2}-1)$	$\frac{n}{2}+1$	$\frac{n}{2}-1$
	Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n}{2}-2) \geq i \geq 0$	$\frac{n}{2}$	$\frac{n}{2}$

Table A2.4: Maximum Quotient Head Lengths.
$n_a = 1, d(x) = x^2+1$

n	Quotient	Head
5	11010	2
7	1110101	3
9	111101010	4
11	11111010101	5

(a) n = odd, $tq_{(n-1)}$ included, or $tq_{(n-1)}$ excluded and secondary selection = tq_i,

$$(n-1) \ge i \ge \left(\frac{n+1}{2}\right)$$

n	Quotient	Head
5	11101	3
7	1111010	4
9	111110101	5
11	11111101010	6

(b) n = odd, $tq_{(n-1)}$ excluded, with secondary selection = tq_i,

$$\left(\frac{n-1}{2}\right) \ge i \ge 0$$

n	Quotient	Head
6	111010	3
8	11110101	4
10	1111101010	5
12	111111010101	6

(c) n = even, $tq_{(n-1)}$ excluded, or $tq_{(n-1)}$ included and secondary selection = tq_i,

$$\left(\frac{n}{2}-2\right) \ge i \ge 0$$

n	Quotient	Head
6	110101	2
8	11101010	3
10	1111010101	4
12	111110101010	5

(d) n = even, $tq_{(n-1)}$ included, with secondary selection = tq_i,

$$(n-1) \ge i \ge \left(\frac{n}{2}-1\right)$$

Table A2.5: Quotients which exhibit $H_{n,max}$

$n_a = 1, d(x) = x^2+1$

Continuing in this manner, it can be shown that the error rate prior to expurgation is:

$$Pe_{\text{partially decoded}} = 2\,Pe_{\text{channel}} - 2\sum_{i=1}^{\text{Int}[^{(N-1)}/_{2}]} i\,Pe_{\text{channel}}^{(i+1)}\,(1-Pe_{\text{channel}})^{i}\ ,$$

where N is the number of transmitted bits. Expurgation of every nth bit results in the probability that $^{1}/_{n}$ of the errors will be removed. The remaining $[^{(n-1)}/_{n}]^{th}$ of the errors will be distributed over $^{(n-1)}/_{n}$ bits. Therefore, the decoded error rate will not change as a result of expurgation, and:

$$Pe_{\text{continuously decoded}} = 2\,Pe_{\text{channel}} - 2\sum_{i=1}^{\text{Int}[^{(N-1)}/_{2}]} i\,Pe_{\text{channel}}^{(i+1)}\,(1-Pe_{\text{channel}})^{i}\ .$$

For realistic bit error rates, the probability of two or more errors occurring within a short span is negligible. Therefore:

$$Pe_{\text{continuously decoded}} \approx 2\,Pe_{\text{channel}}\ .$$

The agreement between the full and approximate expressions for continuously decoded error extension is shown in Figure A2.2.

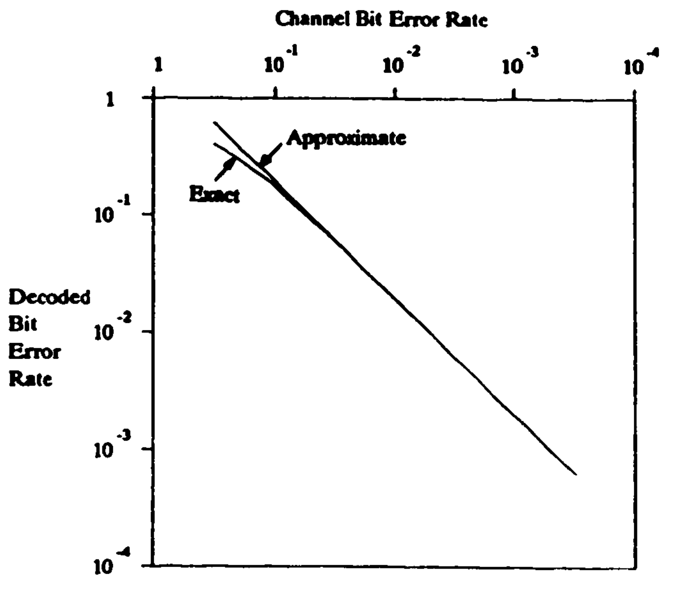


Figure A2.2: Actual vs. Approximate Decoded Error Rate.
Continuous Decoding, $n_s = 1$, $d(x) = x^2 + 1$

Block Decoding:

If decoding is performed on a word by word basis, extension of errors is guaranteed not to extend over word boundaries. When $n_a = 1$, $d(x) = x^2+1$, a single transmission error will cause a single decoded error if the errored bit is the least significant, second least significant, or the most significant bit of a word. Errors in the other (n-3) bit positions of each word will multiply to two decoded errors. Following expurgation, these errors will be distributed over $^{(n-1)}/_{n}$ bits. If the probability of occurrence of two or more errors spaced one bit apart is ignored, then:

$$Pe_{\text{block decoded}} \approx \frac{[2(n-3)+3]}{n} \frac{n}{(n-1)} Pe_{\text{channel}}$$

$$\approx \left[\frac{2n-3}{n-1}\right] Pe_{\text{channel}} .$$

<u>A2.1.3 Quotient Selection and Performance Summary</u>

Since performance in terms of maximum consecutive like bits and error extension is identical for all quotient selection mechanisms, choice of the mechanism can only be based on encoded stream transition density and circuit complexity. As shown in Table A2.1, the greatest value of td_{min} for odd n is ensured when $to_{(n-1)}$ is included in the transition count. Inclusion of $to_{(n-1)}$ also means that a single quotient will be selected in every possible selection set. Since the values of td_{min} for all mechanisms are equal when n is even, the simplest decision mechanism can be used. This mechanism excludes $to_{(n-1)}$ from the transition count.

The performance of this GS code with $n_a = 1$, $d(x) = x^2+1$ when the above encoder decision mechanisms are used is given in Table A2.6.

n	Selection Mechanism	L_{max}	td_{min}	Approximate Error Extension	Efficiency
odd	Select the quotient with the greatest number of transitions. Include $to_{(n-1)}$ in transition count.	n	$\frac{1}{2} + \frac{1}{2n}$	Continuous decoding: 2	$\frac{n-1}{n}$
even	Select the quotient with the greatest number of transitions. Exclude $to_{(n-1)}$ from transition count.	n	$\frac{1}{2}$	Block decoding: $\frac{2n-3}{n-1}$	

Table A2.6: Guided Scrambling Performance Summary:
$n_a = 1$, $d(x) = x^2 + 1$

A2.2 $n_a = 1, d(x) = x+1$

In addition to the performance metrics analyzed for $n_a = 1$, $d(x) = x^2+1$, the possibility for balanced encoding with $d(x) = x+1$ allows for analysis of encoded stream WRDS and DSV bounds. Following the derivation of optimal encoder selection mechanisms, the performance of this GS configuration will be given. Only analysis which varies in methodology from that of the previous section is outlined in detail.

A2.2.1 Quotient Selection Mechanism

Selection of a quotient which exhibits a word-end running digital sum which falls within WRDS bounds ensures balanced transmission. Clearly, selection of the quotient which exhibits minimum $|WRDS|$ would enforce these bounds. When quotients exhibit equal $|WRDS|$ values, selection of the quotient with the preceding transition ensures that only one quotient will always be selected. Alternatively, selection of the quotient with the greatest number of transitions which exhibits a WRDS which falls within WRDS bounds will result in the transmission of as many transitions as possible, given that transmission is balanced.

When $d(x) = x+1$ and augmentation is with a single bit per word, the two quotients offered for selection are complementary and contain transitions in the same opportunities. Since the most significant bits of the quotients differ in value, only one of the quotients will contain a transition prior to the first bit. It is this quotient which will have the greatest number of transitions associated with it. The selection mechanism which ensures the highest encoded stream transition density given balanced transmission and this GS configuration reduces to the selection of the quotient with a transition preceding the first bit, unless selection of this quotient would cause violation of WRDS bounds. The performance of this configuration is considered here. Since performance analysis of the code which selects the quotient with minimum $|WRDS|$ is similar, its performance is considered in the section summary only.

A2.2.2 Performance Analysis

In order to establish the DSV bound which is characteristic of balanced transmission, it is first necessary to establish the WRDS bounds. Following this derivation, it becomes possible to derive L_{max} and a lower bound for td_{min}. Error extension is also considered.

A2.2.2.1 Minimum WRDS bounds

Given the selection between complementary quotients, a situation can arise where selection must be made between quotients composed wholly of like bits. The maximum word disparity for transmitted quotients is then:

$$|WD|_{max} = n .$$

The minimum enforceable WRDS bounds are established by realizing that provision must be made for the possibility of such a selection given that the WRDS prior to quotient selection is at the center of the RDS range, c. This gives:

$$\text{WRDS bounds}_{min} = c \pm n \ .$$

A2.2.2.2 Minimum DSV bound

If the minimum enforceable encoded stream RDS bounds can be established, the minimum stream DSV immediately follows. Consider then the derivation of RDS_{max}. Evaluation of RDS_{min} is symmetrical.

The value below which stream RDS cannot be restrained can be found by considering the following situation:

(1) the WRDS prior to quotient selection is $WRDS_{max}$,

(2) and the selected quotient contains the largest possible string of 1's at the start of the word while exhibiting:

$$WD = \begin{cases} -1, & n = \text{odd} \\ -2, & n = \text{even} \ . \end{cases}$$

From Equation (A2.1a), the maximum number of ones and thus the maximum head length possible in a quotient with this disparity is found to be:

$$H_{max} = \begin{cases} \dfrac{n-1}{2}, & n = \text{odd} \\ \dfrac{n}{2} - 1, & n = \text{even} \ , \end{cases}$$

and:

$$RDS_{max} = WRDS_{max} + H_{max}$$

$$= \begin{cases} c + \left(\dfrac{3n-1}{2}\right), & n = \text{odd} \\ c + \left(\dfrac{3n}{2} - 1\right), & n = \text{even} \ . \end{cases}$$

Similarly:

$$RDS_{min} = \begin{cases} c - \left(\dfrac{3n-1}{2}\right), & n = \text{odd} \\ c - \left(\dfrac{3n}{2} - 1\right), & n = \text{even} \ . \end{cases}$$

Therefore:

$$DSV_{min} = \tfrac{1}{2}(RDS_{max} - RDS_{min})$$

$$= \begin{cases} \left(\frac{3n-1}{2}\right), & n = \text{odd} \\ \left(\frac{3n}{2} - 1\right), & n = \text{even} . \end{cases}$$

A2.2.2.3 Maximum Consecutive Like Bits

Since the derivation of the longest possible run of consecutive ones is symmetrical to that for zeros, only the case of consecutive ones will be considered in this analyis.

It is clear that a word composed solely of ones might be selected for transmission, given its presence in the selection set. It is also clear, given the decision mechanisms of Section A2.2.1, that consecutive "all one" words would not be selected. A string of consecutive like bits can therefore span at most three successive quotient selections, as depicted in Figure A2.3.

Consider the situation which causes a string of consecutive ones. In the absence of WRDS bounds, a quotient can always be selected with a transition prior to its first bit. To cause selection of Word(B) and Word(C), selection of the complement of Word(B) would have to violate $WRDS_{min}$ and selection of the complement of Word(C) would have to violate $WRDS_{max}$.

Consider now an increase in RDS bounds. As RDS_{max} is increased, violation of $WRDS_{max}$ by the complement of Word(C) requires an increase in its disparity. Word(C) would correspondingly have decreased disparity. As the WD of Word(C) decreases, so does the value of $H_{C,max}$, resulting in a decrease in the overall maximum consecutive run of ones. As the distance between RDS_{min} and RDS_{max} is further increased, it becomes possible for Word(C) to always have a zero in the most significant bit position. A lower bound for L_{max} can be established through the addition of $T_{A,max}$ and n.

Since WRDS bounds play a role in determining L_{max}, evaluation of maximum like bits will be considered when:

$$\left.\begin{aligned} WRDS_{max} &= c + (n + k) \\ WRDS_{min} &= c - (n + k) \end{aligned}\right. \quad , \; k \geq 0 .$$

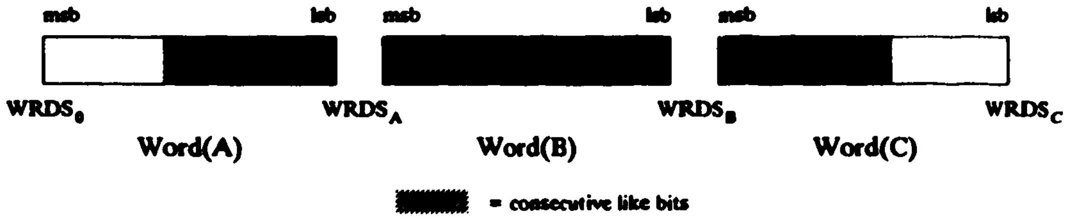


Figure A2.3: Situation for maximum consecutive like bits.
$n_a = 1, d(x) = x + 1$

It should be noted that, when n is even, the even nature of WD values dictates that WRDS bounds will be even also. Consider the above equations. For even WRDS bounds when n is even and k is odd, the center of the RDS range must be an odd number, resulting in WRDS bounds which cannot be equal in magnitude. This non-symmetrical RDS range does not result in unbalanced transmission but merely a slight increase in the required encoder selection mechanism circuitry.

To determine L_{max}, consider the conditions required for its generation and the resulting values of $T_{A,max}$ and $H_{C,max}$.

Condition which forces selection of the "all one" word for Word(B), given its membership in the quotient selection set:

In order to force the selection of the "all one" word given that the previously transmitted word had a mark as its least significant bit:

$$\begin{aligned} WRDS_A &< WRDS_{min} + n \\ &< c - k \, . \end{aligned}$$

For n = odd, word disparity values are odd. When n = even, WD values vary in steps of two. Therefore:

$$WRDS_{A,max} = \begin{cases} c - (k+1) \, , & n = \text{odd} \\ c - (k+2) \, , & n = \text{even} \, . \end{cases}$$

Evaluation of $T_{A,max}$:

The maximum tail length of 1's on Word(A) will occur when the disparity of Word(A) is a maximum. Accordingly:

$$\begin{aligned} WD_A &\leq WRDS_{A,max} - WRDS_{min} \\ WD_{A,max} &= n - 2 \, . \end{aligned}$$

Insertion of this value of $WD_{A,max}$ into Equation (A2.1a) yields:

$$T_{A,max} = n - 1 \, .$$

It is easy to find examples of words with this length of tail which would be selected in this situation.

Given this length of tail on Word(A) and the occurrence of the "all one" word as Word(B):

$$\begin{aligned} WRDS_B &= WRDS_{A,max} + n \\ &= \begin{cases} (c+n) - (k+1) \, , & n = \text{odd} \\ (c+n) - (k+2) \, , & n = \text{even} \, . \end{cases} \end{aligned}$$

Condition which forces selection of a quotient with a head of 1's for Word(C), given its membership in the quotient selection set:

To force selection of a word with a head of 1's given that the previously selected word also ended with a 1 and violation of $WRDS_{min}$ is not possible, the complement of the selected word must violate $WRDS_{max}$. Therefore, when C′ denotes the complement of Word(C):

$$WD_{C'} > WRDS_{max} - WRDS_{n}$$

$$> \begin{cases} 2k + 1, & n = \text{odd} \\ 2k + 2, & n = \text{even} \end{cases},$$

and:

$$WD_{C} < \begin{cases} -(2k + 1), & n = \text{odd} \\ -(2k + 2), & n = \text{even} \end{cases},$$

$$WD_{C,max} = \begin{cases} -(2k + 3), & n = \text{odd} \\ -(2k + 4), & n = \text{even} \end{cases}.$$

Evaluation of $H_{C,max}$:

The maximum head length of 1's in Word(C) will occur when the disparity of this word is a maximum. Given the values of $WD_{C,max}$ calculated above, Equation (A2.1a) yields:

$$H_{C,max} = \begin{cases} \dfrac{n - (2k + 3)}{2}, & n = \text{odd} \\ \dfrac{n - (2k + 4)}{2}, & n = \text{even} \end{cases}.$$

Once again it is simple to show that words with such head lengths would be selected in this situation.

Since the value of $H_{C,max}$ must be greater than or equal to zero, k is limited in value. For $H_{C,max} \geq 0$, simple arithmetic yields:

$$k \leq \begin{cases} \dfrac{n-3}{2}, & n = \text{odd} \\ \dfrac{n}{2} - 2, & n = \text{even} \end{cases}.$$

Combining this limit for k with its original definition, the above expression for $H_{C,max}$ is valid when:

$$0 \leq k \leq \begin{cases} \dfrac{n-3}{2}, & n = \text{odd} \\ \dfrac{n}{2} - 2, & n = \text{even} \end{cases}.$$

Evaluation of L_{max} :

Combining the values for $T_{A,max}$, n, $H_{C,max}$ and k, L_{max} is found to be:

n = odd:

$$L_{max} = \begin{cases} \left(\frac{5n-5}{2}\right) - k \,, & 0 \leq k \leq \left(\frac{n-3}{2}\right) \\ (2n-1) \,, & k > \left(\frac{n-3}{2}\right) \end{cases}$$

n = even:

$$L_{max} = \begin{cases} \left(\frac{5n}{2} - 3\right) - k \,, & 0 \leq k \leq \left(\frac{n}{2} - 2\right) \\ (2n-1) \,, & k > \left(\frac{n}{2} - 2\right) \,, \end{cases}$$

where the corresponding values of WRDS and DSV bounds are, for k ≥ 0:

$$\text{WRDS bounds} = c \pm (n + k) \,,$$

$$\text{DSV} = \begin{cases} \left(\frac{3n-1}{2}\right) + k \,, & n = \text{odd} \\ \left(\frac{3n}{2} - 1\right) + k \,, & n = \text{even} \,. \end{cases}$$

Since there is a point where further expansion of encoded stream WRDS and DSV bounds no longer offers a reduction in L_{max}, there is no advantage in increasing these bounds past:

$$\text{WRDS bounds} = \begin{cases} c \pm \left(\frac{3n-3}{2}\right) \,, & n = \text{odd} \\ c \pm \left(\frac{3n}{2} - 2\right) \,, & n = \text{even} \,, \end{cases}$$

$$\text{DSV} = \begin{cases} (2n-2) \,, & n = \text{odd} \\ (2n-3) \,, & n = \text{even} \,. \end{cases}$$

A2.2.2.4 Minimum transition density

The derivation of a lower bound for encoded stream transition density is straightforward. Noting that successive L_{max} like bit sequences cannot occur because the conditions required to initiate such a sequence are not present immediately following this sequence:

$$td_{min} > \frac{1}{L_{max}} \,.$$

A2.2.2.5 Error Extension

Following the reasoning given in Section A2.1.2.3, it is quite straightforward to establish that, when N bits have been transmitted:

$$Pe_{\text{continuously decoded}} = 2\,[\,Pe_{\text{channel}} - \sum_{i=1}^{N} (\,i\,Pe_{\text{channel}}^{\,(i+1)}\,)\,]$$

$$\approx 2\,Pe_{\text{channel}}\ .$$

Since n_e is less than or equal to the degree of d(x), it is expected that the performance of block and continuous decoding will be the same. This is indeed the case. When only single errors within a transmitted word are considered and decoding is on a word by word basis, multiplication by $d(x) = x+1$ maps errors in the most significant and least significant codeword bit positions to single errors in the decoded word. Errors in the other (n-2) bits are multiplied to two errors. Since only $^{(n-1)}/_{n}$ of the bits remain following stream expurgation:

$$Pe_{\text{block decoded}} \approx \frac{[2(n-2)+2]}{n}\,\frac{n}{(n-1)}\;Pe_{\text{channel}}$$

$$\approx 2\,Pe_{\text{channel}}\ .$$

A2.2.3 Performance Summary

A summary of Guided Scrambling performance for $n_e = 1$, $d(x) = x+1$ when the selected quotient contains a preceding transition wherever enforcement of WRDS bounds permits is given in Table A2.7. As previously mentioned, an alternate selection mechanism could select the word with minimum |WRDS|. Since this mechanism would enforce the tightest WRDS bounds possible, DSV and L_{max} values for that selection mechanism equal those for k = 0 in Table A2.7.

A2.3 $n_e = 1$, $d(x) \in D_{mod6}(x)$

To repeat the polynomials which constitute the family $D_{mod6}(x)$ previously given in Table 4.2:

$$d(x) = \begin{cases} x^{n-1} + x^{n-2} + x^{n-4} + x^{n-6} + x^{n-7} + \ldots + 1, & n \bmod 6 = 0 \\ x^{n-2} + x^{n-4} + x^{n-5} + x^{n-7} + x^{n-8} + \ldots + x + 1, & n \bmod 6 = 2 \\ x^{n-1} + x^{n-3} + x^{n-4} + x^{n-6} + x^{n-7} + \ldots + x + 1, & n \bmod 6 = 4\ . \end{cases}$$

The portrayal of some of these polynomials in bit sequence form in Table A2.8 clearly demonstrates their pattern. Polynomial weight is included in this table for future reference.

As found in Section 4.2.3, these polynomials can produce balanced transmission only when the quotient length is even. Accordingly, only even n will be considered in the following analysis. A discussion of options for the encoder decision mechanism will be followed by a brief performance analysis and summary.

n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
odd	**Select the quotient with transition in $tq_{(n-1)}$, given enforcement of WRDS bounds.**	$c \pm (n+k)$, $0 \leq k \leq (\frac{n-3}{2})$	$(\frac{3n-1}{2}) + k$, $0 \leq k \leq (\frac{n-3}{2})$	$(\frac{5n-5}{2}) - k$, $0 \leq k \leq (\frac{n-3}{2})$	$> \frac{1}{L_{max}}$	**Block or Continuous decoding:** 2	$\frac{n-1}{n}$
		$\geq c \pm \frac{3}{2}(n-1)$	$\geq (2n-2)$	$2n-1$			
even	**Select the quotient with transition in $tq_{(n-1)}$, given enforcement of WRDS bounds.**	$c \pm (n+k)$, $0 \leq k \leq (\frac{n}{2}-2)$	$(\frac{3n}{2}-1) + k$, $0 \leq k \leq (\frac{n}{2}-2)$	$(\frac{5n}{2}-3) - k$, $0 \leq k \leq (\frac{n}{2}-2)$			
		$\geq c \pm (\frac{3n}{2}-2)$	$\geq (2n-3)$	$2n-1$			

Table A2.7: Guided Scrambling Performance Summary:
$n_s = 1$, $d(x) = x + 1$

n	d(x) (msb ... lsb)	weight of d(x)
4	1011	3
6	101101	4
8	1011011	5
10	1011011011	7
12	101101101101	8
14	1011011011011	9

Table A2.8: Bit Sequence Representation of Polynomials from family $d(x) \in D_{mod}(x)$.

A2.3.1 Quotient Selection Alternatives

A decision mechanism which selects the word with the most transitions, given that the word selected does not violate WRDS bounds, should take into account that the quotients will differ only in the first three transition opportunities. The quotients can differ in transition count by at most three, and if the transition count encompasses the three initial opportunities, there is no possibility for indecision.

Since the most significant bits of the two quotient sequences are guaranteed to differ in value, an alternate simple decision mechanism could select the quotient with the preceding transition. Since this method of selection is not concerned with the total number of transitions associated with each quotient, it is expected that the encoded stream transition density will be slightly lower than that for the previously mentioned mechanism. However, it will be shown that for certain WRDS bounds, this mechanism offers a reduction in L_{max}.

A2.3.2 Performance Analysis

Although performance analysis follows the same lines as that for $d(x) = x+1$, several differences must be detailed. These differences arise because the two quotients are no longer complementary but have bits of equal value in their second most significant bit positions.

A2.3.2.1 Minimum WRDS bounds

As derived in Appendix 1, when quotients are related through modulo-2 addition with a sequence which does not complement a single bit position:

$$\#ones_1 = \#zeros_0 \pm 1$$
$$\#zeros_1 = \#ones_0 \mp 1 ,$$

where the subscripts 0 and 1 refer to the sequences $q_0(x)$ and $q_1(x)$ respectively. Given this relationship, it is clear that in all quotient selection sets:

$$WD_1 = -(WD_0 \pm 2) .$$

The (+) sign holds if the non-inverted bit is a space, the (-) sign if the non-inverted bit is a mark. Since the maximum word disparity is n, it is clear then that in every quotient selection set:

$$|WD_0| \text{ or } |WD_1| \leq (n-2) \ .$$

Therefore, the minimum WRDS bounds which can be enforced are:

$$\text{WRDS bounds}_{min} = c \pm (n-2) \ .$$

A2.3.2.2 Minimum DSV bound

To establish the minimum DSV bound, the minimum enforceable value for RDS_{max} must be found. This can be derived by considering the following situation:

(1) WRDS prior to quotient selection is $WRDS_{max}$.

(2) Quotient selection must be from a set which contains words with WD = +2 and WD = 0.

The word with WD = 0 must be selected. The maximum string of ones which can appear at the start of the word, as given by Equation (A2.1a), is:

$$H_{max} = \frac{n}{2} \ .$$

It is easy to show that such selection sets exist. For example, consider the following eight bit selection set:

$$q_0(x) : 01001111 \qquad WD = 2$$
$$q_1(x) : 11110000 \qquad WD = 0 \ .$$

$q_1(x)$ would have to be selected if the WRDS prior to quotient selection was $WRDS_{max}$. Accordingly, the minimum value of RDS_{max} which can be enforced is:

$$RDS_{max} = WRDS_{max} + H_{max}$$
$$= c + \left(\frac{3n}{2} - 2\right) \ .$$

Due to the symmetry in establishing RDS_{min}, it is straightforward to show that:

$$DSV_{max} = \frac{3n}{2} - 2 \ .$$

A2.3.2.3 Maximum Consecutive Like Bits

Since this GS configuration allows for the selection of a quotient containing bits of one polarity, consecutive like bits can span three words in the manner previously diagramed in Figure A2.3. To determine L_{max} then, it is necessary to determine $T_{A,max}$ and $H_{C,max}$. Since expansion of WRDS bounds will once again decrease L_{max}, the variable k will be introduced to allow variation of these bounds. The WRDS bounds then considered are:

$$\text{WRDS bounds} = c \pm [\,(n-2)+k\,], \qquad k \geq 0\ .$$

In the manner described in the analysis of d(x) = x+1, it is quite straightforward to find that when the encoder decision mechanism selects the quotient with the greater number of transitions:

$$T_{A,max} = n-2,$$

$$H_{C,max} = \begin{cases} (\frac{n}{2}-k), & 0 \leq k \leq (\frac{n}{2}-1) \\ 1, & k \geq (\frac{n}{2}-1)\ . \end{cases}$$

However, when $k \geq (^n/_2 - 1)$, selection of the quotient with the preceding transition will reduce $H_{C,max}$ to zero. $H_{C,max}$ then becomes:

$$H_{C,max} = \begin{cases} (\frac{n}{2}-k), & 0 \leq k \leq (\frac{n}{2}-2) \\ 0, & k \geq (\frac{n}{2}-1)\ . \end{cases}$$

L_{max} is derived through simple addition of $T_{A,max}$, n, and $H_{C,max}$. When the quotient with the most transitions is selected, given that it does not violate WRDS bounds, L_{max} is:

$$L_{max} = \begin{cases} (\frac{5n}{2}-2)-k, & 0 \leq k \leq (\frac{n}{2}-1) \\ (2n-1), & k \geq (\frac{n}{2}-1)\ . \end{cases}$$

When the quotient with the preceding transition is selected, given enforcement of WRDS bounds, L_{max} is:

$$L_{max} = \begin{cases} (\frac{5n}{2}-2)-k, & 0 \leq k \leq (\frac{n}{2}-2) \\ (2n-2), & k \geq (\frac{n}{2}-1)\ . \end{cases}$$

4.2.3.2.4 Minimum encoded stream transition density

As in previous derivation, a lower bound for encoded stream transition density is:

$$td_{min} > \frac{1}{L_{max}}\ .$$

4.2.3.2.5 Error Extension

Continuous Decoding:

When the transmission error rate is low enough that the probability of two errors occurring within the space of a word is negligible:

$$Pe_{\text{continuously decoded}} \approx [\text{weight of } d(x)] * Pe_{\text{channel}} .$$

The pattern of d(x) displayed in Table A2.8 indicates that the weight of d(x) is:

$$\text{weight of } d(x) = \left(\frac{n}{2} + 1\right) + \text{Int}\left(\frac{n-4}{6}\right) ,$$

where Int denotes the integer portion of its argument. Therefore, for continuous decoding:

$$Pe_{\text{continuously decoded}} \approx \left[\left(\frac{n}{2} + 1\right) + \text{Int}\left(\frac{n-4}{6}\right)\right] Pe_{\text{channel}} .$$

Block Decoding:

Clearing the decoding multiplication register during word by word decoding will truncate the error extension. When the probability of two or more errors within the span of a word is ignored, the probability of an error in each bit position of the word is assumed $^1/_n$, the number of decoded errors resulting from a transmission error in each bit position is summed, and provision is made for the expurgation of the most significant bit of the word, it is found that:

$$Pe_{\text{block decoded}} \approx \frac{1}{(n-1)} \left[-1 + \sum_{j=1}^{R} j \left[\text{Int}\left(\frac{j+3}{2}\right) - \text{Int}\left(\frac{j}{2}\right) \right] \right] Pe_{\text{channel}} ,$$

where:

$$R = \begin{cases} \frac{2n}{3} , & n \bmod 6 = 0 \\ \frac{2n-1}{3} , & n \bmod 6 = 2 \\ \frac{2n+1}{3} , & n \bmod 6 = 4 . \end{cases}$$

A2.3.3 Performance Summary

A summary of the performance for Guided Scrambling when $n_s = 1$, $d(x) \in D_{\text{mod6}}(x)$ is given in Table A2.9. A selection mechanism which selects the word with minimum |WRDS| once again would exhibit performance corresponding to k = 0 in this table.

A2.4 $n_s = 1, d(x) = x^{n-2} + x^{n-3} + 1$

Analysis of Guided Scrambling performance when the scrambling polynomial is $d(x) = x^{n-2}+x^{n-3}+1$ follows closely that of the previous section. Notable differences occur only in the value of DSV bound and the encoder decision mechanism which allows best performance in terms of $L_{\max}$. Consequently, only these aspects will be discussed.

n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
even	Select the quotient with the most transitions, given enforcement of WRDS bounds. Need consider first three transition opportunities only.	$c \pm (n-2+k)$, $0 \leq k \leq (\frac{n}{2}-1)$	$(\frac{3n}{2}-2)+k$, $0 \leq k \leq (\frac{n}{2}-1)$	$(\frac{5n}{2}-2)-k$, $0 \leq k \leq (\frac{n}{2}-1)$	$> \frac{1}{L_{max}}$	Continuous decoding: $[(\frac{n}{2}+1)+\mathrm{Int}(\frac{n-4}{6})]$ Block decoding: $(\frac{1}{n-1})\left[-1+\sum_{i=1}^{R} i\,[\mathrm{Int}(\frac{i+3}{2})-\mathrm{Int}(\frac{i}{2})]\right]$ $R = \begin{cases} \frac{2n}{3}, & n \bmod 6 = 0 \\ \frac{2n-1}{3}, & n \bmod 6 = 2 \\ \frac{2n+1}{3}, & n \bmod 6 = 4 \end{cases}$	$\frac{n-1}{n}$
		$\geq c \pm (\frac{3n}{2}-3)$	$\geq (2n-3)$	$2n-1$			
	Select the quotient with transition in $to_{(n-1)}$, given enforcement of WRDS bounds.	$c \pm (n-2+k)$, $0 \leq k \leq (\frac{n}{2}-2)$	$(\frac{3n}{2}-2)+k$, $0 \leq k \leq (\frac{n}{2}-2)$	$(\frac{5n}{2}-2)-k$, $0 \leq k \leq (\frac{n}{2}-2)$			
		$\geq c \pm (\frac{3n}{2}-4)$	$\geq (2n-3)$	$2n-2$			

Table A2.9: Guided Scrambling Performance Summary: $n_a = 1$, $d(x) \in D_{mod6}(x)$

2.4.1 Minimum DSV bound

DSV bounds which are tighter that those when $d(x) \in D_{\max}(x)$ can be enforced. During that analysis, $H_{C,\max}$ was found to be:

$$H_{C,\max} = \frac{n}{2} .$$

In this case, a shorter head length can be enforced.

As in the previous configuration, it is straightforward to show that:

$$WD_1 = -(WD_0 \pm 2) ,$$

where the (+) sign holds if the non-inverted bit is a space, and the (-) sign holds if the non-inverted bit is a mark. If the selection set is to consist of two words, one with WD = +2 and the other with WD = 0, a mark must be in the non-inverting bit position. For this d(x), the non-inverting bit position is the second least significant bit. Therefore, even though Equation (A2.1a) indicates that ($^n/_2$) ones appear in the word with WD = 0, only ($^n/_2$ - 1) of these ones can appear at the start of the word. For example, when n = 8, $H_{C,\max}$ is 3, shown by $q_1(x)$ in the following selection set:

$q_0(x)$:	00011111	WD = 2
$q_1(x)$:	11100010	WD = 0 .

Since $H_{C,\max}$ is reduced by one, so too is the DSV bound. The minimum bound is then:

$$DSV_{\min} = \frac{3n}{2} - 3 .$$

It should be noted that enforcement of WRDS bounds is not sufficient to enforce this DSV bound. Particular care must be taken to ensure that the quotients which exceed RDS bounds but do not violate WRDS bounds are not selected.

A2.4.2 Quotient Selection Criteria and $L_{\max}$

As in preceding analysis of $L_{\max}$, it is clear that consecutive like bits can span at most three words and the maximum tail length of the first word is (n-2). Also, the maximum head length for the third word will vary with the encoder selection mechanism. Because the transition opportunities in which the quotient transitions differ are the one preceding the word and the last two, selection of the word with the greater number of transitions given enforcement of WRDS bounds allows $H_{C,\max}$ to increase as the bounds are increased. Conversely, if a decision is made simply for the word with the preceding transition, $H_{C,\max}$ will decrease with increasing RDS bounds. $L_{\max}$ will increase or decrease in accordance with $H_{C,\max}$.

When the quotient with the greatest number of transitions is selected while WRDS and DSV bounds are enforced, the maximum consecutive like bit run length is:

$$L_{max} = \begin{cases} (\frac{5n}{2}-3)+k, & 0 \le k \le (\frac{n}{2}-1) \\ (3n-4), & k \ge (\frac{n}{2}-1). \end{cases}$$

When the quotient with a transition at $to_{(n-1)}$ is selected, given that it does not violate WRDS bounds, L_{max} is:

$$L_{max} = \begin{cases} (\frac{5n}{2}-3)-k, & 0 \le k \le (\frac{n}{2}-1) \\ (2n-2), & k \ge (\frac{n}{2}-1). \end{cases}$$

In light of these results, and with the realization that reduction in L_{max} is more significant that a slight increase in td_{min}, the latter encoder decision mechanism is recommended.

A2.4.3 Performance Summary

Table A2.10 presents a summary of the line code performance for Guided Scrambling which is configured with $n_a = 1$, $d(x) = x^{n-2}+x^{n-3}+1$, using an encoder decision mechanism which selects the word with a transition at $to_{(n-1)}$ given that this quotient does not violate WRDS or DSV bounds. Values of performance metrics which are presented but not discuss... above were determined by previously outlined methods. Performance reported in this table for enforcement of the tightest WRDS bounds also results when the quotient with minimum |WRDS| is selected.

A2.5 $n_a = 1$, $d(x) = x^{n-1} + x^{n-2} + 1$

The results of Guided Scrambling performance when $n_a = 1$, $d(x) = x^{n-1}+ x^{n-2}+1$, as evaluated in accordance with the methods outlined in the previous section, are summarized in Table A2.11.

A2.6 $n_a = 2$, $d(x) = x^2 + 1$

Analysis of the performance of a Guided Scrambling configuration which involves augmentation of each source word with two bits is much more complicated than that for a GS code which has single bit augmentation. The increase in difficulty is due to the fact that when all patterns of augmenting bits are introduced into each source word, each quotient selection set consists of four quotients rather than just two. As shown previously in Table 4.1, when the two most significant bits are the augmenting bits, the four quotients are related by the sequences:

$$\begin{aligned} r_0(x) &= 0000\ldots00 \\ r_1(x) &= 0101\ldots01 \\ r_2(x) &= 1010\ldots10 \\ r_3(x) &= 1111\ldots11\ . \end{aligned}$$

n	Selection Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
even	Select the quotient with a transition in $to_{(n-1)}$, given enforcement of WRDS and DSV bounds.	$c \pm (n-2+k)$, $0 \leq k \leq (\frac{n}{2}-1)$	$(\frac{3n}{2}-3)+k$, $0 \leq k \leq (\frac{n}{2}-1)$	$(\frac{5n}{2}-3)-k$, $0 \leq k \leq (\frac{n}{2}-1)$	$> \frac{1}{L_{max}}$	Continuous decoding: 3	$\frac{n-1}{n}$
		$\geq c \pm (\frac{3n}{2}-3)$	$\geq (2n-4)$	$2n-2$		Block decoding: $\frac{2n}{n-1}$	

Table A2.10: Guided Scrambling Performance Summary:
$n_a = 1$, $d(x) = x^{n-2} + x^{n-3} + 1$

n	Decision Mechanism	WRDS bounds	DSV bound	L_{max}	td_{min}	Approximate Error Extension	Efficiency
even	Select the quotient with a transition in $to_{(n-1)}$, given enforcement of WRDS and DSV bounds.	$c \pm (n-2+k)$, $0 \leq k \leq (\frac{n}{2}-1)$	$(\frac{3n}{2}-3)+k$, $0 \leq k \leq (\frac{n}{2}-1)$	$(\frac{5n}{2}-3)-k$, $0 \leq k \leq (\frac{n}{2}-1)$	$> \frac{1}{L_{max}}$	Continuous decoding: 3	$\frac{n-1}{n}$
		$\geq c \pm (\frac{3n}{2}-3)$	$\geq (2n-4)$	$2n-2$		Block decoding: $\frac{2n-1}{n-1}$	

Table A2.11: Guided Scrambling Performance Summary:
$n_a = 1$, $d(x) = x^{n-1} + x^{n-2} + 1$

Corresponding to the increased size of the quotient selection set is an improvement in performance.

Discussion of the encoder selection mechanisms which might be used is followed by analysis of the performance in light of these mechanisms. Evaluation of the most appropriate mechanisms and a summary of performance concludes the analysis.

A2.6.1 Quotient Selection Alternatives

The use of two augmenting bits with $d(x) = x^2+1$ can ensure balanced transmission and additionally allow for optimization of a second line code characteristic. Accordingly, quotient selection solely in terms of minimum $|\mathrm{WRDS}|$ will not be considered. The secondary characteristic which will be considered is the transmission of a high number of level transitions.

To ensure balanced transmission while at the same time providing for a high encoded stream transition density, the word with the most transitions which does not violate WRDS bounds can be selected. However, as the discussion of the encoder decision mechanism options for $n_a = 1$, $d(x) = x^2+1$ might indicate, quotient selection cannot be quite that straightforward. This mechanism does not ensure that only one quotient will be selected in each situation. Table A2.12 outlines situations in which this single selection mechanism is not sufficient. If arbitrary selection between quotients with an equal number of transitions is not desired, a secondary selection mechanism must be introduced. It will be shown in the following analysis that these secondary selection mechanisms can affect line code performance.

n	Transition count includes $to_{(n-1)}$.	Transition count exludes $to_{(n-1)}$.
odd	$q_0(x)$ and $q_2(x)$ or $q_1(x)$ and $q_3(x)$ might simultaneously have $t = (\frac{n-1}{2})$ or $t = (\frac{n+1}{2})$	All quotients can simultaneously contain $t_{im} = (\frac{n-1}{2})$ transitions
even	$q_0(x)$ and $q_1(x)$ or $q_2(x)$ and $q_3(x)$ might simultaneously have $t = (\frac{n}{2})$	$q_0(x)$ and $q_3(x)$, and $q_1(x)$ and $q_2(x)$ will always contain the same number of transitions.

Table A2.12: Situations requiring further quotient selection.
$n_a = 2$, $d(x) = x^2+1$

Considering the situations in which a secondary mechanism might play a part in selecting a quotient and the secondary selection mechanisms which can easily and adequately select such a quotient, the following quotient selection mechanisms will be considered during performance analysis:

n = odd:

Select the quotient with the greatest number of transitions, given that it does not violate WRDS or DSV bounds, including $to_{(n-1)}$ in the transition count. In the event that two quotients are judged equal, select the quotient which contains a transition at a predetermined to_i, $(n-2) \geq i \geq 0$. Appropriate selection of i will be considered in the following analysis.

n = even:

Select the word with the greatest number of transitions which does not violate WRDS or DSV bounds. In the event that two or more quotients are equal in this respect:

(1) where the transition count has included $to_{(n-1)}$, make further quotient selection by selecting the word with a transition at to_i, $(n-1) \geq i \geq 0$. The appropriate value of i must be determined.

(2) where $to_{(n-1)}$ has been excluded from the transition count, select the quotient with a transition at $to_{(n-1)}$.

A2.6.2 Performance Analysis

A2.6.2.1 Encoded stream transition density

During analysis of the GS code which used $d(x) = x^2+1$ with single bit augmentation, it was found that a quotient with $t_{int,min}$ transitions could always be selected, where:

$$t_{int,min} = \begin{cases} \dfrac{n-1}{2}, & n \text{ odd} \\[2ex] \dfrac{n}{2}, & n \text{ even}. \end{cases}$$

This number of transitions was assured since the quotients were related by an alternating sequence. To derive td_{min} for the present GS configuration, note:

(1) the four member quotient selection set is composed of two pairs of complementary words: $q_0(x)$ and $q_3(x)$, $q_1(x)$ and $q_2(x)$. Complementary words have an equal number of transitions within them.

(2) the quotients related by the addition of an alternating sequence are:

$q_0(x)$ and $q_1(x)$ $\qquad$ $q_1(x)$ and $q_3(x)$

$q_0(x)$ and $q_2(x)$ $\qquad$ $q_2(x)$ and $q_3(x)$

At least one of the words in each of these pairs must have $t_{int} \geq t_{int,min}$.

quotient	selection set configuration (a)	selection set configuration (b)	selection set configuration (c)
$q_0(x)$	$\geq t_{int,min}$	$< t_{int,min}$	$\geq t_{int,min}$
$q_1(x)$	$< t_{int,min}$	$\geq t_{int,min}$	$\geq t_{int,min}$
$q_2(x)$	$< t_{int,min}$	$\geq t_{int,min}$	$\geq t_{int,min}$
$q_3(x)$	$\geq t_{int,min}$	$< t_{int,min}$	$\geq t_{int,min}$

Table A2.13: Number of transitions within quotients for each possible quotient selection set configuration.
$n_a = 2, d(x) = x^2+1$

In accordance with these observations, Table A2.13 presents the only conditions which can arise within the quotient selection set. It is clear from this table that there will always be the possibility of selecting between complementary words which have $t_{int} \geq t_{int,min}$. Selection between complementary words ensures that transmission can be balanced.

In the case of single bit augmentation and odd n, lack of WRDS bounds allowed for the selection of a quotient which also contained a transition prior to their most significant bit. It will be shown in the following analysis that enforcement of WRDS bounds at times removes this option. It will also be shown that situations which disallow selection of words with a preceding transition cannot occur repetitively. Lower bounds for encoded stream transition density are therefore given by:

$$td_{min} > \frac{t_{int,min}}{n}$$

$$> \begin{cases} \frac{1}{2} - \frac{1}{2n}, & n \text{ odd} \\ \frac{1}{2}, & n \text{ even} . \end{cases}$$

Selection of words with fewer than $t_{int,min}$ transitions will improve other performance metrics only when n mod 4 = 2. This is discussed in Section A2.6.2.3.

A2.6.2.2 Minimum WRDS bounds

When the preceding values of $t_{int,min}$ are used in Equation (A2.2), values for selected quotient maximum disparity are found. Table A2.14 summarizes these findings. It is quite easy to find quotient selection sets which are composed entirely of words which have word disparities greater than or equal to these values of $|WD|_{min}$. Table A2.15 gives examples of such sets for n = 8 through 11. Given that quotient selection must be from one of these selection sets when the WRDS prior to selection is at the center of the running digital sum range, it would be impossible to enforce tighter

n	$t_{\text{int,min}}$	$\|WD\|_{max}$
n mod 4 = 0	$\frac{n}{2}$	$\frac{n}{2}$
n mod 4 = 1	$\frac{n-1}{2}$	$\frac{n+1}{2}$
n mod 4 = 2	$\frac{n}{2}$	$\frac{n}{2}-1$
n mod 4 = 3	$\frac{n-1}{2}$	$\frac{n-1}{2}$

Table A2.14: Selected Quotient $|WD|_{max}$. $n_a = 2$, $d(x) = x^2+1$

n	Possible Selection Set	WD	$\|WD\|_{max}$ from Table A2.14
8	$q_0(x)$: 00001010 $q_1(x)$: 01011111 $q_2(x)$: 10100000 $q_3(x)$: 11110101	-4 4 -4 4	4
9	$q_0(x)$: 000001010 $q_1(x)$: 010100000 $q_2(x)$: 101011111 $q_3(x)$: 111110101	-5 -5 5 5	5
10	$q_0(x)$: 0000010101 $q_1(x)$: 0101000000 $q_2(x)$: 1010111111 $q_3(x)$: 1111101010	-4 6 -6 4	4
11	$q_0(x)$: 00000010101 $q_1(x)$: 01010111111 $q_2(x)$: 10101000000 $q_3(x)$: 11111101010	-5 5 -5 5	5

Table A2.15: Quotient Selection Sets from which quotients exhibiting $|WD|_{max}$ must be selected. $n_a = 2$, $d(x) = x^2 + 1$.

WRDS bounds than:

$$\text{WRDS bounds}_{min} = c \pm \begin{cases} \frac{n}{2}, & n \bmod 4 = 0 \\ \frac{n+1}{2}, & n \bmod 4 = 1 \\ \frac{n}{2} - 1, & n \bmod 4 = 2 \\ \frac{n-1}{2}, & n \bmod 4 = 3\ . \end{cases}$$

A2.6.2.3 Minimum DSV bounds

As in previous analysis, a bound on encoded stream digital sum variation can be established through addition of WRDS_{max} and the longest head of 1's possible in a word which has negative word disparity and contains at least $t_{int,min}$ transitions. Addition of minimum WRDS bounds to values for H_{max} derived from Equation (A2.4) suggests that:

$$\text{DSV}_{min} = \begin{cases} \frac{3n-4}{2}, & n \bmod 4 = 0 \\ \frac{3n+1}{2}, & n \bmod 4 = 1 \\ \frac{3n-6}{2}, & n \bmod 4 = 2 \\ \frac{3n-1}{2}, & n \bmod 4 = 3\ . \end{cases}$$

Although these bounds can be enforced, when n mod 4 = 3 and when td_{min} is allowed to fall below the value determined in Section A2.6.2.2 for n mod 4 = 2, the DSV bounds can be tighter.

Consider the case of n mod 4 = 3. Given that a word must contain $t_{int,min} = {}^{(n-1)}/_{2}$ transitions, Equation (A2.4a) finds that the longest possible value for H_{max} when WD = -1 is ${}^{(n+1)}/_{4}$. It is simple to find words which match this description, such as $q_3(x)$ in the following seven bit quotient selection set.

$q_0(x)$:	0011011	WD = 1	$t_{int} = 3$
$q_1(x)$:	0110001	WD = - 1	$t_{int} = 3$
$q_2(x)$:	1001110	WD = 1	$t_{int} = 3$
$q_3(x)$:	1100100	WD = - 1	$t_{int} = 3$.

Addition of this value of H_{max} to WRDS_{max} yields the previously mentioned value for DSV_{min}.

But, consider once again the above quotient selection set. $q_1(x)$ also has a negative word disparity and contains as many transitions within the word as $q_3(x)$, yet exhibits a word-internal

RDS_{max} one less than $q_3(x)$. It can be shown that for any n, n mod 4 = 3, such a selection will always be available. Without a reduction in td_{min}, DSV bounds can be tightened by 1 so that:

$$DSVmin = \frac{3n-5}{4}, \qquad n \bmod 4 = 3 .$$

A similar situation arises when n mod 4 = 2. The value of H_{max} calculated from Equation (A2.4a) is $^{(n-2)}/_{4}$. $q_3(x)$ in the following ten bit quotient selection set is an example of a word which exhibits this length of head.

$q_0(x)$:	0011010111	WD = 2	$t_{int} = 5$
$q_1(x)$:	0110000010	WD = - 4	$t_{int} = 4$
$q_2(x)$:	1001111101	WD = 4	$t_{int} = 4$
$q_3(x)$:	1100101000	WD = - 2	$t_{int} = 5$.

If a word with $^n/_2 = 5$ transitions must be chosen from this selection set, $q_3(x)$ would be selected. Note however that while containing one less internal transition, $q_1(x)$ has a negative disparity and a word internal RDS_{max} one less than $q_3(x)$. It can be shown that for any n, n mod 4 = 2, such a selection is always available. Recognizing that a tighter bound on encoded stream DSV is of greater importance than a slight increase in transition density, the minimum DSV bound for n mod 4 = 2 is given as:

$$DSV_{min} = \frac{3n-10}{4}, \qquad n \bmod 4 = 2 ,$$

where td_{min} is now:

$$td_{min} > \frac{1}{2} - \frac{1}{n}, \qquad n \bmod 4 = 2 .$$

Summarizing the minimum DSV bounds:

$$DSV_{min} = \begin{cases} \frac{3n-4}{2}, & n \bmod 4 = 0 \\ \frac{3n+1}{2}, & n \bmod 4 = 1 \\ \frac{3n-10}{2}, & n \bmod 4 = 2 \\ \frac{3n-5}{2}, & n \bmod 4 = 3 . \end{cases}$$

A2.6.2.4 Maximum Consecutive Like Bits

Derivation of L_{max} for n mod 4 = 0 through n mod 4 = 3, taking each secondary decision technique mentioned in Section A2.6.1 into account, is quite long and tedious. Consequently, only the methodology of procedure will be outlined, followed by tabular presentation of the results of a

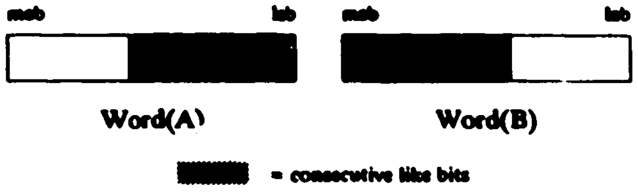


Figure A2.4: Situation for maximum consecutive like bits.
$n_a = 2, d(x) = x^2+1$

complete analysis.

As in the case for $n_a = 1$, $d(x) = x^2+1$, the requirement for the occurrence of a minimum number of transitions within each transmitted word indicates that the "all one" or "all zero" word will never be selected. A string of consecutive like bits can therefore span portions of at most two words as shown previously in Figure A2.1 and repeated here for convenience as Figure A2.4.

Since there is the opportunity to select between complementary words, in the absence of WRDS bounds a quotient which contains a transition at $to_{(n-1)}$ will always be selected. It is only due to the enforcement of a WRDS bound that a quotient without a transition at $to_{(n-1)}$ would be selected. Since the numerical value of the bound enforced and the existence of the other bound play no part in the extension of this like bit sequence, variation of WRDS bounds will not change to its length. Therefore, variation of WRDS bounds will not be considered in the following analysis.

To determine the maximum possible length of consecutive bits, the following situations were considered:

(1) Like bits = 1, Bound enforced is $WRDS_{max}$ } symmetrical
Like bits = 0, Bound enforced is $WRDS_{min}$

(2) Like bits = 1, Bound enforced is $WRDS_{min}$ } symmetrical .
Like bits = 0, Bound enforced is $WRDS_{max}$

Maximum head and tail lengths in each case were established through:

(1) use of Equation (A2.4) to determine maximum values of T_{max} and H_{max} given enforcement of WRDS bounds and the number of transitions required within each word.
(2) consideration of any reduction in T_{max} and H_{max} resulting from enforcement of DSV bounds.
(3) consideration of the secondary decision mechanism options and their provision for selection of a word other than that which exhibits T_{max} or H_{max}, given that the word containing T_{max} or H_{max} is available for selection.

The results of this analysis is presented in Tables A2.16 through A2.19. Table A2.20 summarizes the selection mechanisms which result in the shortest run length of consecutive like bits and the corresponding values of L_{max} for all n.

Selection Mechanism	Bound Enforced / Polarity of consecutive like bits	T_{max}	H_{max}	L	L_{max}
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(n-1) \ge i \ge (\frac{n}{2}-1)$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n}{2}$	$\frac{n-4}{4}$	$\frac{3n-4}{4}$	n - 2
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	$\frac{n-2}{2}$	$\frac{n-2}{2}$	n - 2	
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n}{2}-2) \ge i \ge 0$ or Exclude $to_{(n-1)}$ from transition count. If counts equal, select the quotient with a transition at $to_{(n-1)}$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n}{2}$	$\frac{n-4}{4}$	$\frac{3n-4}{4}$	n - 1
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	$\frac{n-2}{2}$	$\frac{n}{2}$	n - 1	

Table A2.16: T_{max}, H_{max}, L_{max} when n mod 4 = 0, n_a = 2, $d(x) = x^2+1$

Selection Mechanism	Bound Enforced / Polarity of consecutive like bits	T_{max}	H_{max}	L	L_{max}
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(n-2) \ge i \ge (\frac{n+1}{2})$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n+1}{2}$	$\frac{n-1}{4}$	$\frac{3n+1}{4}$	4, n = 5 n - 2, n > 5
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	$\frac{n-3}{2}$	$\frac{n}{2}$	n - 2	
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n-1}{2}) \ge i \ge 0$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n-1}{2}$	$\frac{n-1}{4}$	$\frac{3n-3}{4}$	n
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	$\frac{n-1}{2}$	$\frac{n+1}{2}$	n	

Table A2.17: T_{max}, H_{max}, L_{max} when n mod 4 = 1, n_a = 2, $d(x) = x^2+1$

Selection Mechanism	Bound Enforced / Polarity of consecutive like bits	T_{max}	H_{max}	L	L_{max}
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(n-1) \geq i \geq (\frac{n}{2} - 1)$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n}{2}$	$\frac{n-6}{4}$	$\frac{3n-6}{4}$	8, $n = 10$ $n - 1$, $n > 10$
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	3, $n = 10$ $\frac{n-2}{2}$, $n > 10$	$\frac{n}{2}$	8, $n = 10$ $n - 1$, $n > 10$	
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n}{2} - 2) \geq i \geq 0$ or Exclude $to_{(n-1)}$ from transition count. If counts equal, select the quotient with a transition at $to_{(n-1)}$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n}{2}$	$\frac{n-2}{4}$	$\frac{3n-2}{4}$	8, $n = 10$ $n - 1$, $n > 10$
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	3, $n = 10$ $\frac{n-2}{2}$, $n > 10$	$\frac{n}{2}$	8, $n = 10$ $n - 1$, $n > 10$	

Table A2.18: T_{max}, H_{max}, L_{max} when n mod 4 = 2, n_s = 2, $d(x) = x^2 + 1$

Selection Mechanism	Bound Enforced / Polarity of consecutive like bits	T_{max}	H_{max}	L	L_{max}
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(n-2) \geq i \geq (\frac{n+1}{2})$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n+1}{2}$	$\frac{n-3}{4}$	$\frac{3n-1}{4}$	$n - 1$
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	$\frac{n-1}{2}$	$\frac{n-1}{2}$	$n - 1$	
Include $to_{(n-1)}$ in transition count. If counts equal, select the quotient with a transition at to_i, $(\frac{n-1}{2}) \geq i \geq 0$	$WRDS_{max}$ / 1's or $WRDS_{min}$ / 0's	$\frac{n-1}{2}$	$\frac{n-3}{4}$	$\frac{3n-5}{4}$	$n - 1$
	$WRDS_{min}$ / 1's or $WRDS_{max}$ / 0's	$\frac{n-1}{2}$	$\frac{n-1}{2}$	$n - 1$	

Table A2.19: T_{max}, H_{max}, L_{max} when n mod 4 = 3, n_s = 2, $d(x) = x^2 + 1$

n	Secondary Selection*	L_{max}
n mod 4 = 0	Select quotient with transition in to_i, $(n-1) \geq i \geq (\frac{n}{2}-1)$	n - 2
n mod 4 = 1	Select quotient with transition in to_i, $(n-2) \geq i \geq (\frac{n+1}{2})$	4, n = 5 n - 2, n > 5
n mod 4 = 2	Select quotient with transition in any predetermined transition opportunity.	8, n = 10 n - 1, n > 10
n mod 4 = 3	Select quotient with transition in to_i, $(n-2) \geq i \geq 0$	n - 1

* Primary Selection Mechanism: Select quotient with most transitions that does not violate WRDS or DSV bounds. Inlude $to_{(n-1)}$ in transition count.

Table A2.20: Summary of shortest L_{max} and their decision mechanisms. $n_a = 2$, $d(x) = x^2 + 1$

A2.6.2.5 Error Extension

Error extension for $n_a = 2$, $d(x) = x^2+1$ closely follows that for $n_a = 1$, $d(x) = x^2+1$, resulting in:

$$\begin{aligned} Pe_{decoded} &= 2\left[Pe_{channel} - \sum_{i=1}^{Int[(N-1)/2]} i\, Pe_{channel}^{(i+1)} (1-Pe_{channel})^{i} \right] \\ &\approx 2\, Pe_{channel}, \end{aligned}$$

independent of whether decoding is done in a block or continuous manner.

A2.6.3 Performance Summary

A summary of the best decision mechanisms and corresponding Guided Scrambling performance when $n_a = 2$, $d(x) = x^2+1$ is given in Table A2.21.

n	Secondary Selection*	WRDS bounds	DSV bound	L_{max}	td_{min}	Error Extension	Efficiency
n mod 4 = 0	Select quotient with transition in to_i, $(n-1) \geq i \geq (\frac{n}{2}-1)$	$\frac{n}{2}$	$\frac{3n-4}{4}$	n - 2	$> \frac{1}{2}$	Block or Continuous decoding: 2	$\frac{n-2}{n}$
n mod 4 = 1	Select quotient with transition in to_i, $(n-2) \geq i \geq (\frac{n+1}{2})$	$\frac{n+1}{2}$	$\frac{3n+1}{4}$	4, n = 5 n - 2, n > 5	$> (\frac{1}{2} - \frac{1}{2n})$		
n mod 4 = 2	Select quotient with transition in any predetermined transition opportunity.	$\frac{n-2}{2}$	$\frac{3n-10}{4}$	8, n = 10 n - 1, n > 10	$> (\frac{1}{2} - \frac{1}{n})$		
n mod 4 = 3	Select quotient with transition in to_i, $(n-2) \geq i \geq 0$	$\frac{n-1}{2}$	$\frac{3n-5}{4}$	n - 1	$> (\frac{1}{2} - \frac{1}{2n})$		

*Primary Selection Mechanism: Select quotient with most transitions that does not violate WRDS or DSV bounds. Inlude $tq_{(n-1)}$ in transition count.

Table A2.21: Guided Scrambling Performance Summary:
$n_s = 2,\ d(x) = x^2 + 1$

APPENDIX 3

GUIDED SCRAMBLING SPECTRAL ANALYSIS

This appendix introduces the procedure for block coded signal spectral analysis derived by Cariolaro and Tronca [24], and outlines the technique by which this method can be applied to Block and Continuous Guided Scrambling.

A3.1 Spectra of Block Coded Digital Signals

Cariolaro and Tronca [24] have investigated spectral analysis of constant length block coded digital signals. Assuming only that the input word sequence is wide-sense stationary and the input words are statistically independent, they have extended fundamental spectral analysis principles to calculate the influence that block encoding rules have on the spectra of their encoded digital sequences. This section briefly outlines their derivation.

For a stochastic process X(t) to be wide-sense stationary, it must be characterized by a mean value which is independent of time and an autocorrelation $\phi(t_1,t_2)$ that depends not on the specific time instants t_1,t_2 but on their difference [27]. This autocorrelation can be written:

$$\begin{aligned} \phi(t_1,t_2) &= E\,[X(t_1),X(t_2)] \\ &= \phi(t_1 - t_2) \\ &= \phi(\tau)\ , \end{aligned} \tag{A3.1}$$

where E denotes evaluation of expectation and $\tau = (t_1 - t_2)$. A cyclostationary process is one which is periodically stationary in the wide sense.

The distribution of power with frequency for a stochastic process is computed by taking the Fourier transform of the autocorrelation function:

$$\Phi(f) = \int_{-\infty}^{\infty} \phi(\tau)\, e^{-j2\pi f\tau}\, d\tau\ . \tag{A3.2}$$

When considering the block encoded digital sequence, Cariolaro and Tronca demonstrate that stationarity of the input sequence implies only stationarity of the sequence of transmitted codewords, not the transmitted symbol sequence. The encoded symbol sequence is then a cyclostationary process with a period equal to the codeword duration. Evaluation of the encoded sequence discrete autocorrelation becomes, in matrix notation:

$$\phi_k = E\,[tx_i^{T}, tx_{i+k}]\ , \qquad -\infty < i, k < \infty\ , \tag{A3.3}$$

where tx_i is the row vector representation of the codeword $tx_i(x)$ and ^{T} denotes transposition. The power spectral density of the coded signal is then given by:

$$\Phi(f) = f_n\ |S(f)|^2 \sum_{k=-\infty}^{\infty} e^{-j\omega k nT}\, V\, \phi_k\, V^{*}\ , \tag{A3.4}$$

where:

$S(f)$	=	the Fourier transform of the transmitted symbol
T	=	duration of each symbol
n	=	length of the encoded word
f_n	=	$1/nT$
ω	=	$2\pi f$
V	=	the row vector $[e^{j\omega T}, e^{j2\omega T}, e^{j3\omega T}, \ldots, e^{jn\omega T}]$
V^*	=	the transpose conjugate of V .

By separating the continuous spectral component $X_c(f)$ from the discrete components (with weighting function X_d), Equation (A3.4) can be rewritten:

$$\Phi(f) = f_n \, |S(f)|^2 \, \{ X_c(f) + f_n X_d(f) \sum_{k=-\infty}^{\infty} \delta(f - kf_n) \} \, , \qquad \text{(A3.5)}$$

Cariolaro and Tronca then proceed to outline a computationally efficient method for calculation of both $X_c(f)$ and $X_d(f)$, given that the encoder can be modelled as a finite state machine. The required input consists of only the probability of occurrence for each input word and, for each input word in each encoder state, the transmitted codeword and the next state of the encoder. This information is easily obtained when the code rules are expressed in the form of code tables as for the Alphabetic 7B8B block code [10].

The program LCODEPWR, written in double-precision FORTRAN, calculates both the continuous and discrete spectral components for the line code whose parameters have been specified in the file LCODEPWR.IN. Comparison of program results for the Franaszek MS-43 code with Figure 2 of [24] validates program operation.

Appendix 4 contains more information regarding the programs and their use.

A3.2 Spectra of Block GS Codes

Calculation of the power spectral density of Block GS codes is straightforward when the analysis procedure of Cariolaro and Tronca is followed. Evaluation of the probability of occurrence of each m-bit source word is elementary when the input symbols are assumed statistically independent. With this assumption:

$$P[s^j(x)] = p^{\#ones_j} (1-p)^{m-\#ones_j} \, , \qquad 0 \le j \le (2^m-1) \, , \qquad \text{(A3.6)}$$

where:

P	=	probability of occurrence of its argument
p	=	input sequence logic one probability
$\#ones_j$	=	number of ones in $s^j(x)$.

Modelling the encoder as a finite state machine is also straightforward when it is noted that apart from quotient statistics, quotient selection depends only on encoded stream WRDS and the value of the last transmitted bit. In balanced transmission, the encoded stream can take on a finite number of WRDS values. Since these values play a role in codeword selection, each possible WRDS value corresponds to an encoding state. If quotient selection also depends on the value of the last transmitted bit, the number of encoder states doubles to include each value of WRDS when the last transmitted bit is a one and each value of WRDS when the last transmitted bit is a zero. Table A3.1 depicts one possible state assignment for a Block GS7B8B code.

Table A3.2 gives the transmitted codewords and the encoder next states for the same GS7B8B code when the input word is all zero. Complete code spectral analysis input is easily generated by extending such a table to include every encoder state with each possible input word. The program GSPSIN performs this evaluation for GS codes and creates the input file required for line code spectral analysis.

Table A3.1: Block GS7B8B Encoder State Assignment

Last Bit Transmitted	Encoded Stream WRDS	Encoder State Assignment
	+ 8	18
	+ 6	17
	+ 4	16
	+ 2	15
1	0	14
	- 2	13
	- 4	12
	- 6	11
	- 8	10
	+ 8	9
	+ 6	8
	+ 4	7
	+ 2	6
0	0	5
	- 2	4
	- 4	3
	- 6	2
	- 8	1

Table A3.2: Transmitted Quotient and Encoder Next State for all zero input word.

Current State	Transmitted Quotient	Next State
18	00000000	5
17	00000000	4
16	00000000	3
15	00000000	2
14	00000000	1
13	11111111	17
12	11111111	16
11	11111111	15
10	11111111	14
9	00000000	5
8	00000000	4
7	00000000	3
6	00000000	2
5	11111111	18
4	11111111	17
3	11111111	16
2	11111111	15
1	11111111	14

Code Parameters:

$d(x) = x + 1$

Selection mechanism: Select quotient with preceeding transition, given enforcement of WRDS bounds.

WRDS bounds enforced: $\pm$ 8

A3.3 Spectra of Continuous GS Codes

A3.3.1 Continuous GS Encoder Model

If inappropriately modelled, a continuous GS encoder can assume many more states than the corresponding block encoder. When the model extends from source word input to codeword output, the following straightforward state assignment can lead to 2^{d+1} * (#WRDS values) encoder states:

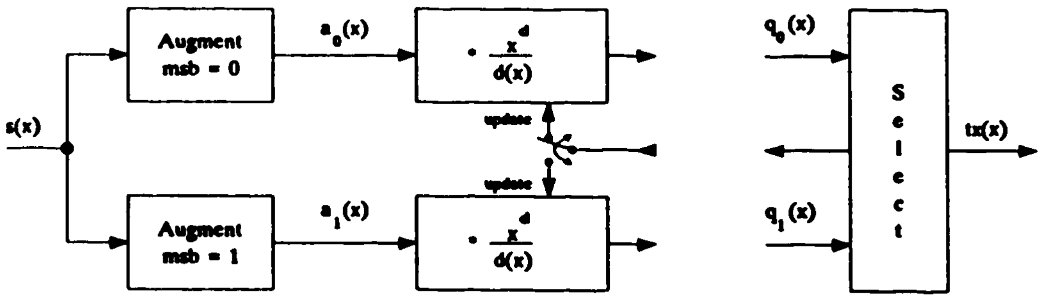
#WRDS values		corresponding to each possible value of encoded stream WRDS
*	2	if the value of the last transmitted bit affects quotient selection
*	2^d	since the division registers will contain one of 2^d possible remainders independent of encoded stream WRDS or value of last bit.

This number of states becomes unmanageable for even small d.

Modelling the encoder in a different fashion reduces the number of states. If the encoder is divided into two processes as shown in Figure A3.1, the second part of the encoder can be modelled as a finite state machine with the same number of states as the Block GS code with the same code parameters. With input $q_0(x)$, knowledge of encoded stream WRDS and the value of the last transmitted bit is sufficient to determine the encoded word and the next state of the encoder. What remains is the evaluation of the probability of occurrence of each $q_0(x)$ sequence.

A3.3.2 Evaluation of $q_0(x)$ probability, $n_a = 1$, $d \leq n$

To evaluate the probability of each $q_0(x)$ sequence when augmentation is with a single bit per word and the degree of the divisor at most equals the encoded word length, consider the formation of $q_0(x)$ depicted in Figure A3.1(a). Let the superscript t denote the sequences currently being encoded. As shown in Appendix 1, $q_0^t(x)$ depends on both $a_0^t(x)$ and $rm^{t-1}(x)$, the remainder associated with the previously selected quotient. With the addition of subscripts, Equations (A1.10) and (A1.11) can be



(a) Quotient generation (b) Quotient selection

Figure A3.1: Two Part Continuous GS Encoder Model, $n_a = 1$.

rewritten:

$$q_0^i(x) = Q_{d(x)}[a_0^{i\prime\prime}(x)\, x^d] \;, \qquad \text{(A3.7)}$$

$$a_0^{i\prime\prime}(x) = a_0^i(x) + [rm^{i-1}(x)\, x^{n-d}] \;. \qquad \text{(A3.8)}$$

Consider now the formation of a particular $q_0^i(x)$, $0 \le i \le (2^n-1)$. Since the mapping from $a_0^{i\prime\prime}(x)$ to $q_0^i(x)$ is one-to-one (see Appendix 1):

$$P[q_0^i(x)] = P[a_0^{i\prime\prime}(x)] \;. \qquad \text{(A3.9)}$$

The probability of occurrence of each $a_0{}'(x)$ sequence can be determined by considering Equation (A3.8) and Figure A3.2, its graphical interpretation. It is clear that a number of $a_0(x)$ and $rm(x)$ sequences can combine to form each $a_0^{i\prime\prime}(x)$ sequence. An $a_0(x)$ sequence which contains the same (n-d) least significant digits as $a_0{}'(x)$ will generate $a_0{}'(x)$ if combined with the appropriate $rm(x)$. Since the first bit of $a_0(x)$ is always zero, there are 2^{d-1} combinations of $a_0(x)$ and $rm(x)$ sequences which form $a_0^{i\prime\prime}(x)$. Denote these sequences with the superscript j.

Since $rm^j(x)$ is the result of previous division, the probability of its occurrence is statistically independent from that of $a_0^j(x)$. Then:

$$P[a_0^{i\prime\prime}(x)] = \sum_{j=1}^{2^{d-1}} P[a_0^j(x)]\, P[rm^j(x)] \;. \qquad \text{(A3.10)}$$

Evaluation of $P[a_0^j(x)]$ is straightforward. Let $s^j(x)$ be the source word from which $a_0^j(x)$ is formed. Assuming statistical independence of the source symbols, the probability of $s^j(x)$ is given by Equation (A3.6). Since the mapping from $s^j(x)$ to $a_0^j(x)$ is one-to-one:

$$P[a_0^j(x)] = P[s^j(x)] \;, \qquad \text{(A3.11)}$$

and can be calculated given source stream logic one probability.

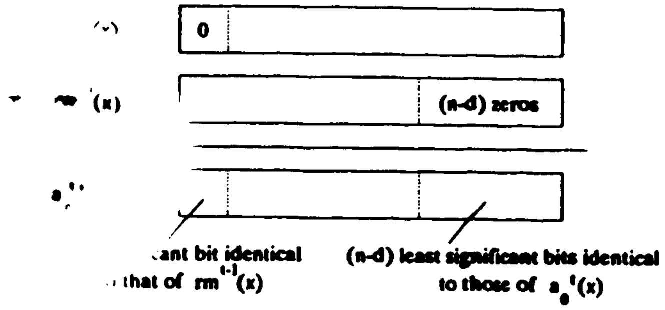


Figure A3.2: Formation of $a_0^{i\prime\prime}(x)$.

Evaluation of $P[rm^j(x)]$ is somewhat more complicated. As derived in Appendix 1, each $rm(x)$ is generated from the division of 2^{n-d} different $a'(x)$ sequences. Let the additional superscript k denote the $a'(x)$ sequences which generate a particular $rm^j(x)$. Since the remainder could result from division of either $a_0^{k\prime}(x)$ or $a_1^{k\prime}(x)$, its probability is given by:

$$P[rm^j(x)] \; = \; \sum_{k=1}^{2^{n-d}} \{ \, P[a_0^{k\prime}(x)] \, Ps[q_0^k(x)] \; + \; P[a_1^{k\prime}(x)] \, Ps[q_1^k(x)] \, \} \; , \qquad \text{(A3.12)}$$

where the mapping from $a_0^{k\prime}(x)$ to $q_0^k(x)$ is one-to-one and Ps denotes the probability that its argument is selected given its presence in the quotient selection set.

Since only the most significant bits of $a_0(x)$ and $a_1(x)$ differ, so too do only the most significant bits of $a_0'(x)$ and $a_1'(x)$. Let $a_0^{k\prime\prime}(x)$ denote the sequence which is generated at the same time as $a_1^{k\prime}(x)$. Mathematically:

$$a_0^{k\prime\prime}(x) \; = \; a_1^{k\prime}(x) + x^{n-1} \; . \qquad \text{(A3.13)}$$

Noting that:

$$Ps[q_1^k(x)] \; = \; 1 - Ps[q_0^k(x)] \; , \qquad \text{(A3.14)}$$

Equation (A3.12) can be rewritten:

$$P[rm^j(x)] \; = \; \sum_{k=1}^{2^{n-d}} \{ \, P[a_0^{k\prime}(x)] \, Ps[q_0^k(x)] \; + \; P[a_0^{k\prime\prime}(x)] \, (1 - Ps[q_0^k(x)]) \, \} \; , \qquad \text{(A3.15)}$$

The probability that each quotient is selected when it is in the quotient selection set is:

$$Ps[q_0^k(x)] \; = \; \sum_{l=1}^{n_s} P[s_l] \, \delta_l[q_0^k(x)] \; , \qquad \text{(A3.16)}$$

where:

$$P[s_l] \; = \; \text{probability that the encoder is in state } l$$

$$n_s \; = \; \text{the number of states in the encoder}$$

$$\delta_l[q_0^k(x)] \; = \; \begin{cases} 1 \, , & \text{if } q_0^k(x) \text{ is selected when encoder is in state } l \\ 0 \, , & \text{otherwise} \; . \end{cases}$$

Values of $\delta_l[q_0^k(x)]$ can easily be determined by examining the codeword selected given input of each $q_0(x)$ in every encoder state.

An expression giving the probability of occurrence of each $q_0^i(x)$ in terms of the probability of occurrence of other $q_0(x)$ sequences, encoder state probabilities and input stream logic one probability can be written by combining (A3.9) - (A3.11) and (A3.15) - (A3.16):

$$P[q_e^i(x)] = \sum_{j=1}^{2^{d-1}} P[s^j(x)] \sum_{k=1}^{2^{n-d}} \{ P[q_e^k(x)] \sum_{l=1}^{n_e} (P[s_l]\, \delta_l[q_e^k(x)]) +$$
$$P[q_e^{k'}(x)]\,(1 - \sum_{l=1}^{n_e} P[s_l]\, \delta_l[q_e^k(x)]) \} \, , \qquad \text{(A3.17)}$$

where the relationships between sequences with superscripts i, j, and k are as before. In matrix form, (A3.17) becomes:

$$\mathbf{p}_q = A\,\mathbf{p}_q \, , \qquad \text{(A3.18)}$$

where:

$\mathbf{p}_q$ = column vector, dimension 2^n, of quotient occurrence probabilities
A = 2^n square coefficient matrix .

With the additional constraint that:

$$\sum_{i=0}^{2^n-1} P[q_e^i(x)] = 1 \, , \qquad \text{(A3.19)}$$

(A3.18) represents a system of linearly dependent equations that can be solved iteratively for a specific value of input stream logic one probability, given values of encoder state probabilities.

In developing their method of block code spectral analysis, Cariolaro and Tronca outline a procedure to solve for encoder state probabilities. To summarize, let $E[q_e^k(x)]$ represent an n_e square binary matrix with element values:

$$e_{q_e^k(x)}(i,j) = \begin{cases} 1 \, , & \text{if the quotient selection mechanism moves from state i to state j following input of } q_e^k(x) \\ 0 \, , & \text{otherwise ,} \end{cases}$$

for $1 \le i, j \le n_e$. With only a change in notation, Equations (22) and (23) from [24] can be written:

$$\mathbf{p}_s = [\, \sum_{k=0}^{2^n-1} P[q_e^k(x)]\, E[q_e^k(x)] \,]^T \mathbf{p}_s \, , \qquad \text{(A3.20)}$$

$$\sum_{l=1}^{n_e} P[s_l] = 1 \, , \qquad \text{(A3.21)}$$

where $\mathbf{p}_s$ is the column vector of encoder state probabilities and ^{T} denotes transposition. The linearly dependent equations of (A3.20), given knowledge of quotient occurrence probabilities and the constraint of (A3.21), can once again be solved through iteration.

Simultaneous iteration of Equations (A3.18) and (A3.20), given the limitations of (A3.19) and (A3.21), will solve for quotient occurrence probabilities. The program Q0PROB performs this iteration. The duplication of spectra calculated for Continuous and Block GS codes with $d \le n_e$ verifies this quotient occurrence probability evaluation.

APPENDIX 4
COMPUTER SIMULATION ENVIRONMENT

Simulation and analysis of Guided Scrambling encoding and decoding procedures has provided validation of Guided Scrambling operation and further characterization of its performance in terms of line code requirements. All simulation and analysis was performed on the IBM PC family of computers. With exception of the line code spectral analysis routine which was written in double precision FORTRAN, all programs were written in Pascal and compiled with the Turbo Pascal 4.0 compiler.

As simulation code was written, it became apparent that programs to simulate the numerous line code configurations and perform the required analysis could be structured in two ways. Complex programs could be written to simulate performance of entire transmission links. The line code configuration and transmission impairments could be altered by replacing segments of the source code. Alternatively, a number of short programs could be written, each simulating only a portion of the transmission link. Information could be transferred between these programs through an underlying set of files. Line code configurations and transmission impairments could be varied by selecting the desired program at each stage of simulation. Code performance could be examined at any point of the link by listing the appropriate file. Because of its flexibility, simplicity of expansion, and ease of use, this latter structure was implemented.

But as the number of programs increased, so too did the difficulty in remembering program names and command line parameters. The creation of an environment which allows for selection and execution of routines from a series of menus removed this drawback. The first section of this appendix describes this versatile and easy to use Pull-Down Menu Shell. The second section outlines utilization of this shell to form the Line Coding Environment.

The complete Line Coding Environment is available from the Alberta Telecommunications Research Centre. Although the environment can be initialized by executing the included LCE batch file, execution speed and convenience improves if it is installed on a hard disk. The included INSTALL routine will copy the environment routines to a hard disk and initialize system variables. The executable code requires 600 kilobytes of disk space for installation. Program source code, if desired, requires an additional 630 kilobytes.

A4.1 The Pull-Down Menu Shell

The pull-down menu shell is a versatile, easy to use environment which can be run on all IBM PC machines and true compatibles. Valid operation has been observed on a number of XT, 286, and 386 machines using CGA, EGA, VGA and Hercules graphics cards. The shell allows users to select previously defined pull-down menu options or to enter and execute any command from a prompt. Up to ten pull-down menus can be defined, each with up to sixteen entries. Selection of a menu entry results in execution of its associated command.

The pull-down menu and environment display options are specified in a definition file prior to shell execution. The structure of this file is discussed in Section A4.1.1. Environment use is outlined in Sections A4.1.2 through A4.1.5 and the code which comprises this shell is briefly mentioned in Section A4.1.6.

A4.1.1 The Definition File

The definition file required for pull-down menu generation must contain information regarding environment frames, prompts and colours, as well as specification of the menu entries and associated commands. The required form and order of these entries is discussed in the following sections.

A4.1.1.1 Form of Entry

Definition file entries can take several forms:

(1) Text entry	:	any printable characters, except curly brackets.
(2) Yes/No entry	:	Y or N, in upper or lower case.
(3) Colour specification	:	an integer between 0 and 15 designating colours as described in Table A4.1.
(4) Prompt information	:	an integer between 0 and 7 requesting display information as outlined in Table A4.2.

Each entry in the definition file must be positioned within curly brackets which cannot extend past the end of a line. Multiple entries per line are allowed. Text outside the brackets is ignored, allowing unlimited comment.

Integer Specification	Result on Colour Monitor	Result on Monochrome Monitor
0	Black	Black
1	Blue	Underlined
2	Green	
3	Cyan	
4	Red	
5	Magenta	
6	Brown	
7	Light Gray	Colour

Adding 8 to the above specifications will intensity colours.

Table A4.1: Definition file Colour Specification

Integer Specification	Information Displayed
0	No information.
1	Current drive.
2	Full directory, including drive.
3	Current subdirectory.
4	Time of day.
5	Time and drive.
6	Time and full directory.
7	Time and current subdirectory.

Table A4.2: Definition file Prompt Specification

A4.1.1.2 File Structure

Specifications in the definition file must be complete, valid, and in the correct order. If the definition file is not properly structured, shell initialization will abort with an appropriate error message. The required order of environment specification entries is as follows:

Environment title.	{ Text, 0 - 70 characters. }
Title colour.	{ Colour specification. }
Main menu frame.	{ y/n }
Main menu frame colour.	{ Colour specification. }
Main menu background colour.	{ Colour specification. }
Main menu text colour.	{ Colour specification. }
Working window frame.	{ y/n }
Working window frame colour.	{ Colour specification. }
Working window background colour.	{ Colour specification. }
Working window text colour.	{ Colour specification. }
Pull-down menu frame.	{ y/n }
Pull-down menu frame colour.	{ Colour specification. }
Pull-down menu background colour.	{ Colour specification. }
Pull-down menu text colour.	{ Colour specification. }
Pull-down menu title colour.	{ Colour specification. }
Cursor prompt information.	{ Prompt specification. }
Cursor prompt colour.	{ Colour specification. }
Corner information.	{ Prompt specification. }
Corner information colour.	{ Colour specification. }
Number of pull down menus.	{ An integer, 1 - 10. } .

And, for each pull-down menu :

Number of entries.	{ An integer, 0 - 16. }	
Title and associated command.	{ Title text, 1 - 26 characters. }	
	{ Command text, 0 - 70 characters. }	
Selections and commands.	{ Entry text 1 }	{ Command 1 }
	{ Entry text 2 }	{ Command 2 }
	:	:
	{ Entry text n }	{ Command n } .

Several items regarding environment specification require further explanation:

(1) The environment title will be truncated if it is longer than 70 characters.

(2) Even when titles, frames, prompts and corner information are not desired, entries for their colour specification are still required. In this instance, null entries are permitted.

(3) Corner information will be written in the lower right hand corner only if the working window is framed.

(4) Since the pull-down menu titles are combined to form the main menu, their collective character count (including a blank character following each title) cannot exceed 77.

(5) Each pull-down menu title and pull-down menu entry must contain at least one non-blank character. They will be truncated if they contain more than 26 characters.

(6) The commands associated with menu selections must be executable and can be at most 70 characters in length.

(7) The command associated with a pull-down menu title will be executed only when the title is selected and the menu does not contain any entries. If the pull-down menu contains entries, a command must still be associated with the title even though it will be ignored. Null entries are permitted.

<u>A4.1.2 Entering the Environment</u>

The pull-down menu shell is entered by executing the program PDMSHELL with syntax:

PDMSHELL [COM=drive:\path\filename] [DEF=drive:\path\filename] [command] ,

where square brackets enclose optional parameters. Explanations for these parameters follow.

To operate, PDMSHELL requires information regarding the location of a DOS command interpreter and the environment definition file. If no command line parameters are issued with the call, these files are searched for under the names:

Command interpreter	: C:\COMMAND.COM ,
Definition file	: PDMSHELL.DEF (in current directory) .

If the command interpreter and/or definition file is in another location or has another name, the following command line parameters can be issued when calling PDMSHELL:

Command interpreter	: COM=drive:\path\filename ,
Definition file	: DEF=drive:\path\filename .

PDMSHELL initially looks for the command interpreter and definition file in the locations specified or in the default locations if parameters are not given. If the definition file cannot be found or is of invalid format, an error message will be given and PDMSHELL execution will terminate. If the command interpreter file is not found, program execution will again terminate with an appropriate error message. As long as a file exists in the specified command interpreter location, PDMSHELL initialization is satisfied. But if this file is not a command interpreter, any attempt at command execution will lock up the machine. It is essential then that the location of the command interpreter be specified correctly.

If the command interpreter is found and the definition file is valid, the environment will be drawn. If the command line holds no more parameters, user input will be awaited. But if execution of a routine is desired immediately upon environment entry, this command can also be issued as a PDMSHELL command line parameter. User input will be awaited upon completion of its execution.

A4.1.3 The Environment

The pull-down environment shell consists of a main menu positioned at the top of the screen, and a working window. The titles of the pull-down menus form the main menu. If a mouse is installed, the mouse cursor will appear in the main menu area. The working window contains a prompt and cursor, awaiting input. User input is accepted from the keyboard, and when installed, a mouse.

Commands can be entered at the cursor. Their execution will be attempted given a carriage return.

Menu items can be selected with either the keyboard or mouse. Depression of the Escape key will toggle the active cursor between the working window and the main menu. Once in the main menu area, the Left and Right arrow keys can be used to select the desired pull-down menu. Depression of the Enter key or Down arrow will cause the pull-down menu to be drawn or will run the command associated with the pull-down menu title if there are no entries in its menu.

If the machine is configured with a mouse, a rectangular mouse cursor will be present in the main menu area. Mouse movement will cause the mouse cursor to move within the main menu. When the mouse cursor is on a pull-down menu title, depression of the left mouse button will cause the pull-down menu to be drawn or will run the command associated with the pull-down menu title if there are no selections within its menu. Note that when the main menu has been entered through an Escape from the keyboard, mouse input is disabled. Escaping back to the working window will re-enable mouse input.

Once in a pull-down menu, a menu entry can be selected. If the menu was entered through keystrokes, Up and Down arrows will select the desired menu option. Depression of the Enter key will run the command associated with the menu selection or an Escape will return action to the main menu. If the pull-down menu was entered through mouse input, the mouse cursor will now be confined to the area of this menu. Depression of the left mouse button will execute the command associated with the menu selection on the row occupied by the mouse cursor. Depression of the right button (or middle button if the mouse has three buttons) will return activity to the main menu.

A4.1.4 Windowed Execution

As mentioned previously, the pull-down menu environment has a main menu area and a working window. This working window includes columns 2 through 78 and rows 4 through 24 of the 80 column, 25 row screen. Although any executable command will run from the shell, only those which confine their input and output to this window area can run within the working window.

PDMSHELL must be informed when a command is capable of confining its operation to these boundaries. There are three ways in which this information can be conferred:

(1) The command is a windowed DOS command. A subset of the internal DOS commands are modified by PDMSHELL to enable windowed operation. Table A4.3 lists these commands. Although these commands are not executed exactly as DOS would prescribe, operation is similar. N. ble differences include:

cls	: Places a clear environment outline on the screen instead of just clearing the working window. It should be used whenever the display becomes distorted.
(x)copy:	Will not notify the user of the progress of multiple file copies until all files have been copied.
exit	: Exits from PDMSHELL, not just most recent command interpreter shell.
print	: Will not run in the background.
type	: Can be issued with the /p switch to type one screen at a time.

(2) The last parameter issued on the command line is "W" or "w". PDMSHELL interprets a final command line parameter of "W" as a cue to the called program to operate within the working window. When this parameter is present, PDMSHELL assumes that the program will interpret it correctly and confine its input and output to the windowed area. The environment display will then be left on the screen. This applies to commands associated with pull-down menu titles and selections as well as those issued from the prompt.

(3) The command entered at the prompt, without the "W" parameter, is associated with a pull-down menu title or selection where it does have a last command line parameter of "W". A command can then by typed quickly and with a number of different parameters without appending "W" each time.

break	erase (del)	set
chdir (cd)	exit	time
cls	mkdir (md)	type
copy	path	ver
date	print	verify
dir	rename (ren)	vol
drive change	rmdir (rd)	xcopy

Table A4.3: Windowed DOS commands

If one of these three techniques is not used to inform PDMSHELL that the command will operate within the working window, the screen will be cleared prior to program execution. Following routine completion, depression of any key will return operation to the pull-down menu environment.

The requirement for operation within the working window and disregard of the final parameter "W" is easily met by programs which have no screen I/O and ignore command line parameters. However, most programs do not fall into this category. For fellow Turbo Pascal buffs, the procedure "pdmwindow", available in the unit "pdm", fulfills the windowing requirements. It looks for the parameter "W" on the end of the command line and if the parameter is present, windows program operation and modifies the "paramcount" system variable so that this parameter is ignored by the rest of the program. It should be the first call in the program, as in:

```
uses
  pdm;
begin
  pdmwindow;
```

Programs in other languages can be windowed, but this task is left up to their author.

A4.1.5 Exiting the Environment

There are three ways in which the pull-down menu shell can be exited:

(1) A pull-down menu entry with the associated command "EXIT" can be selected.

(2) The pull-down menu entry entitled "EXIT" can be selected. If this entry has a command associated with it, this command is executed before the environment is abandoned.

(3) The command "EXIT" is entered at the working window prompt. If the pull-down menu entry "EXIT" exists, the command associated with it will be executed before the environment is exited. If this selection does not exist, the environment will simply be abandoned.

A4.1.6 The Code

PDMSHELL is written in Pascal, compiled with the Turbo Pascal 4.0 compiler. Turbo procedures have been used as much as possible, but a few BIOS interrupt calls were required for screen I/O and mouse control. The source code PDMSHELL.PAS is quite short due to the distribution of routines among several units. These include:

PDMGLOBE.PAS	:	Global declarations and routines,
PDMMOUSE.PAS	:	Mouse I/O,
PDMSETUP.PAS	:	Environment initialization,
PDMSCRN.PAS	:	Screen output,
PDMINPUT.PAS	:	User input interpretation.

And as mentioned previously, the unit PDM.PAS contains the windowing procedure "pdmwindow".

A4.2 The Line Coding Environment

The Line Coding Environment consists of a set of line code simulation and analysis routines which have been collected through use of the Pull-Down Menu Shell. Each program either simulates a portion of a transmission link, generates data for analysis, or performs required analysis. Simulation routines model sources, encode source bit streams though various line code rules, impair transmission, and decode the received bit sequences. The encoded bit stream can be analyzed with respect to realization of line code requirements. The source and decoded bit streams can be compared to determine the accuracy of source stream recovery. The influence that line code rules have on the spectra of the transmitted bit stream can also be analyzed with line code spectral analysis routines. A number of programs are available to generate the required spectral analysis input data.

Programs transfer information through a set of files maintained in he current directory. Line code performance and analysis results can be observed simply by listing the appropriate file. These files are discussed in more detail in Section A4.2.1. Programs can be easily executed through selection of a pull-down menu item. The menus and the effect of item selection are detailed in Section A4.2.2. Section A4.2.3 briefly describes each program written for the Line Coding Environment.

A4.2.1 Environment File Structure

Programs which simulate portions of the transmission link must have as input the bit stream just prior to the function they model and must output the modified bit stream. Similarly, analysis routines must be able to locate required data and have an outlet for analysis results. These input and output requirements are met through use of a standard set of files of common structure which must be present or which will be created in the directory of current operation. Expansion of the environment becomes simple when new routines adopt this structured form of data I/O.

Programs which generate or modify bit streams act on files with the extension .BIT. The following self-explanatory files are required as input or are created by line code simulation and analysis routines:

SOURCED.BIT	RECEIVED.BIT
ENCODED.BIT	DECODED.BIT

Each of these files contains information regarding the history of the bit stream. The first fifteen lines are set aside to record sequence history as shown in Table A4.4. Due to the block nature of most coding rules, the bit stream is recorded as a sequence of words. The words and word numbers are written into these files beginning on line sixteen. The word number is right justified in column seven, the word bit sequence record from most significant bit (first bit in time) to least significant bit starting in column ten. Code rules and analysis routines which view the bit stream as a continuous sequence of bits ignore word boundaries.

Line	Information	
1	Directory of creation	
2	File description	
3	Comment	
4	Length of words	
5	Number of words	
6	Source stream characteristics	
7	Channel impairment	
8	Form of line coding	
9	Line code parameters	
10	Scrambling polynomial	
11	FEC Generator polynomial	For future FEC integration
12	Code shortening polynomial	
13 - 15	Comment	

Table A4.4: Bit sequence history information retained in .BIT files

Programs which require or produce data in forms other than the bit stream traversing the transmission link look for anu write to files with the imaginative .IN and .OUT extensions. Data files created for analysis by other programs are given the analysis routine name and the .IN extension. The results of analysis are place in files with the extension .OUT. For example, line code power spectral analysis requires data in the file LCODEPWR.IN and places analysis results in the file LCODEPWR.OUT. Information similar to that retained in .BIT files regarding line code configuration and transmission impairment is also recorded in the first few line of these files.

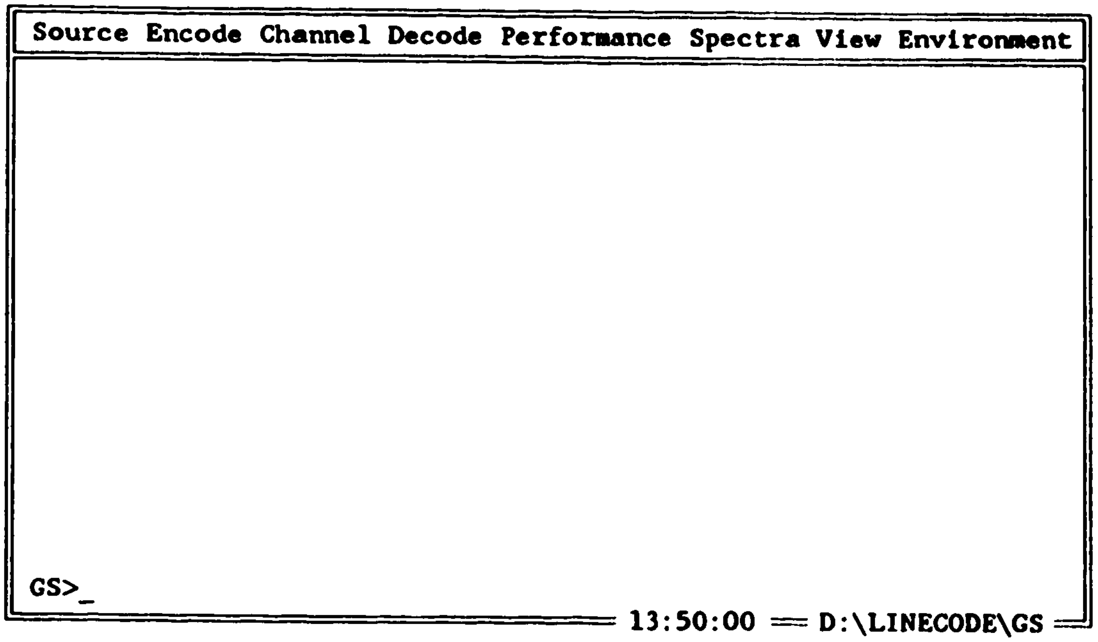
Also, a record is kept of Line Coding Environment routines executed in the current directory. With the exception of the power spectral analysis program, each routine automatically updates a log file with the time, date, and description of analysis or simulation performance. Table A4.5 lists all files which are used by programs in the Line Coding Environment.

A4.2.2 The Environment

The collection of line code simulation and analysis routines into a Pull-Down Menu Shell has resulted in the user-friendly Line Coding Environment. When in use, the environment appearance is similar to that depicted in Figure A4.1. As indicated by the selections in the menu area at the top of the screen, line coding simulation and analysis routines have been grouped into eight functional sets. Each set is accessible from a pull-down menu. The contents of each pull-down menu is displayed in Figure A4.2. Line code simulation and analysis can be performed simply through selection of appropriate menu items. The entry and execution of any DOS command or other programs from keyboard entry complements the versatility of this environment.

File	Contents
SOURCED.BIT	Sourced bit stream
ENCODED.BIT	Encoded bit stream
RECEIVED.BIT	Received bit stream
DECODED.BIT	Decoded bit stream
CHANNEL.OUT	Transmission impairment specifics
CORRECT.OUT	For future integration of error correction
LCODEPRF.OUT	Line code performance results
ERRORPRF.OUT	Error extension performance
LCODEPWR.IN	Power spectral analysis input data
LCODEPWR.OUT	Power spectral analysis results
Q0PROB.OUT	Continuous Guided Scrambling q_0 (x) probabilities
LOG	History of executed LCE routines

Table A4.5: Line Coding Environment files


```
Source Encode Channel Decode Performance Spectra View Environment

GS>_
                                         13:50:00 == D:\LINECODE\GS
```

Figure A4.1: Line Coding Environment Display

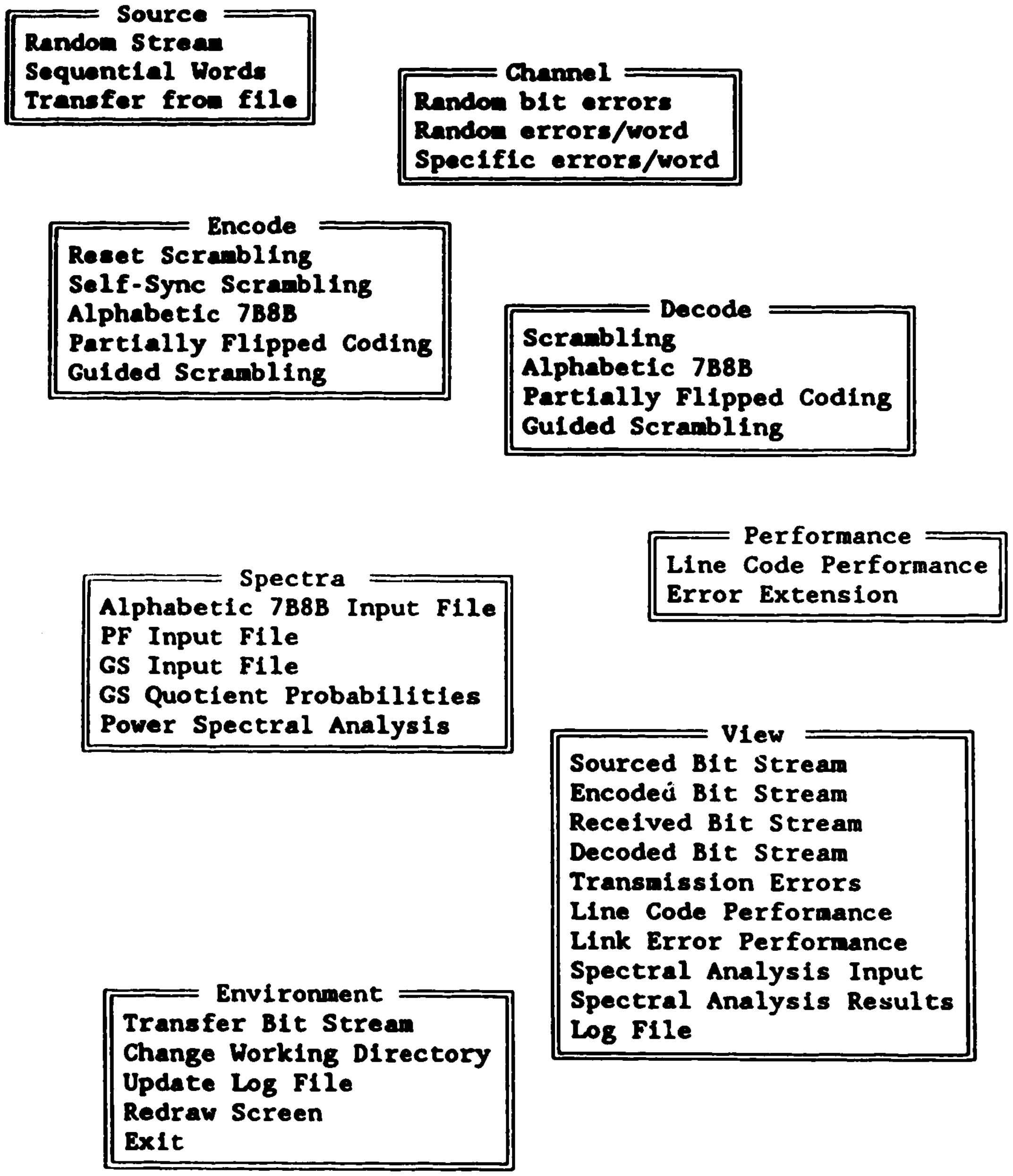


Figure A4.2: Line Coding Environment Pull-Down Menus

A brief discussion of the functionality available from pull-down menu item selection follows. The next section discusses in greater detail the individual programs which comprise the Line Coding Environment.

Source:

Selection of the appropriate entry will generate a source bit stream which is random, a series of sequential words, or simply transferred from another file. Normalized source stream dc (the probability of a logic one in the source stream) is specified by the user when a random bit stream is generated.

Encode:

Item selection offers simulation of a number of line encoding rules. Reset and self-synchronizing scrambling routines view the source bit stream as a continuous sequence. Other line code rules observe the word boundaries in the source bit stream. User interaction with the encoding routines further defines the line code configuration.

Channel:

Corruption of the transmitted bit stream results from selection of Channel pull-down menu items. Error can be introduced into the encoded bit stream randomly in accordance with a user specified bit error rate, randomly within each transmitted word, or in specific bit positions.

Decode:

The received bit stream will be decoded according to the line code configuration parameters which accompany the bit stream when an item is selected from this menu. If channel impairments have not been introduced, the uncorrupted encoded bit stream will be decoded.

Performance:

Item selection initiates analysis of either the performance of the encoding rules with respect to line code requirements or the accuracy of source bit stream recovery.

Spectra:

The input files required for line code spectral analysis can be generated and spectral analysis performed through selection of the appropriate entry from this menu. As derived in Appendix 3, the generation of spectral analysis input for the Continuous Guided Scrambling codes in more complicated than for other line code configurations. For programming convenience, data generation has been divided into two stages. Required spectral input for this code form is completed by evaluation fo the Continuous Guided Scrambling quotient probabilities following GS input file generation.

View:

Each file created by Line Coding Environment routines can be viewed through selection of the appropriate item from this menu. File contents are listed one screen at a time.

Environment:

Item selection accesses miscellaneous environment routines. A bit stream can be transferred from one file to another, with the option of changing the length and number of its words. The directory of current operation can be changed. This routine updates the log file in addition to DOS chdir functionality. Information can also be written into the log file, the environment display can be redrawn, and the environment exited.

A4.2.3 Program Specifics

A brief discussion of each program included in the Line Coding Environment is given in this section. More exhaustive descriptions are available in the introductions prefacing the program source code. Comments throughout executable code segments further detail the structure and operation of each routine.

In the following description, square brackets enclose parameters which can be given when calling the routine. Input and output refers only to file I/O exclusive of log file updates. Routines also interact with the user by writing to the CRT and awaiting input from the keyboard.

Source bit stream generation:

Program :	SOURCE [RANDOM] [SEQUENTIAL] [TRANSFER]
Input :	none
Output :	SOURCED.BIT
Function :	Generates a source bit stream and files it in SOURCED.BIT. A command line parameter can be issued to specify whether the bit stream is to be random, a series of sequential words, or transferred from another file. The [illegible] will be asked to specify sequence form if a parameter is not given as well as th[illegible] normalized dc level for the random stream, word length and number of words.

Simulation of line encoding rules:

Program :	SCRAMBLE [RESET] [SELFSYNC]
Input :	SOURCED.BIT
Output :	ENCODED.BIT
Function :	Encodes the source bit stream through either reset or self-synchronizing scrambling procedures. Scrambling form can be specified with a command line parameter or in response to program query. The scrambling polynomial and original shift register contents must be specified. Scrambling ignores source stream word boundaries.

Program : 7B8BENCODE

Input : SOURCED.BIT

Output : ENCODED.BIT

Function : Performs standard Alphabetic 7B8B encoding by translating a series of 7 bit source words to 8 bit codewords using the look-up tables published by Sharland and Stevenson [10].

Program : PFENCODE

Input : SOURCED.BIT

Output : ENCODED.BIT

Function : Encoded a series of odd length source words with the rules for Partially Flipped codes as published by W.A. Krzymien [17].

Program : GSENCODE [1] [2] [3] [4] [5] [6]

Input : SOURCED.BIT

Output : ENCODED.BIT

Function : Encodes the source bit stream through Guided Scrambling. An integer on the command line is assumed to be the number of augmenting bits desired in each source word. If this parameter is not present, the user will be asked to specify the number of augmenting bits in addition to the scrambling polynomial, quotient selection mechanism, and form of shift register update.

Introduction of transmission errors:

Program : CHANNEL [RANDOMBIT] [RANDOMWORD] [SPECIFICWORD]

Input : ENCODED.BIT

Output : RECEIVED.BIT, [CHANNEL.OUT]

Function : Introduces errors into the encoded bit stream in specific bit positions in each word, random locations in each word, or randomly throughout the encoded bit stream in accordance with a user specified bit error rate. The form of transmission impairment can be specified as a command line parameter or in response to program query. Information regarding the introduced errors can be written into CHANNEL.OUT if desired.

Simulation of line decoding rules:

Program : UNSCRAMBLE

Input : RECEIVED.BIT or ENCODED.BIT

Output : DECODED.BIT

Function : Unscrambles the received bit stream according to the scrambling parameters which accompany the bit stream. If the encoded bit stream has not been corrupted to form the RECEIVED.BIT sequence, the encoded stream will be decoded instead. The user also has the option of respecifying scrambling parameters.

Program :	7B8BDECODE
Input :	RECEIVED.BIT or ENCODED.BIT
Output :	DECODED.BIT
Function :	Decodes the received or encoded bit stream using a look-up table inverse to that used during Alphabetic 7B8B encoding.

Program :	PFDECODE
Input :	RECEIVED.BIT or ENCODED.BIT
Output :	DECODED.BIT
Function :	Decodes the received or encoded bit stream according to the rules for Partially Flipped decoding.

Program :	GSDECODE
Input :	RECEIVED.BIT or ENCODED.BIT
Output :	DECODED.BIT
Function :	Applies Guided Scrambling decoding rules to the received or encoded bit stream. The code parameters which accompany the bit stream can be used during decoding, or the code configuration can be interactively respecified.

Simulation analysis:

Program :	LCODEPRF
Input :	SOURCED.BIT and/or ENCODED.BIT
Output :	LCODEPRF.OUT
Function :	Analyzes the source and/or encoded bit streams with respect to satisfaction of line code requirements. Analysis results can be written out for each word or can be given in summary form

Program :	ERRORPRF
Input :	SOURCED.BIT, DECODED.BIT, [CHANNEL.OUT], [CORRECT.OUT]
Output :	ERRORPRF.OUT
Function :	Compares the source and decoded bit streams to determine the accuracy of source bit stream recovery. If the files CHANNEL.OUT or CORRECT.OUT exist in the current directory, their data will be incorporated into analysis results. Error performance results can be written out on a word by word basis or in summary form.

Line code power spectral analysis and input data generation:

Program :	7B8BPSIN
Input :	none
Output :	LCODEPWR.IN
Function :	Generates the data necessary for line code spectral analysis of the Alphabetic 7B8B code and structures in the format required by the program LCODEPWR. Details of this spectral analysis were given in Appendix 3. The user interactively specifies the source stream dc level and the number of spectral points to be calculated.

Program : PFPSIN

Input : none

Output : LCODEPWR.IN

Function : Generates data required for line code spectral analysis of the Partially Flipped family of codes. Specification of source word length, source stream logic one probability and number of spectral points is requested of the user.

Program : GSPSIN

Input : none

Output : LCODEPWR.IN

Function : Generates data required for Guided Scrambling spectral analysis. The user is asked to specify the GS code configuration in a manner similar to that for Guided Scrambling encoding. Generation of the required data for Block Guided Scrambling configurations is straightforward and is completed by this program. As discussed in Appendix 3, generation of spectral analysis input is more complicated for Continuous Guided Scrambling codes. This program completes only the first portion of spectral analysis data generation for these continuous codes. The following program must be subsequently executed to complete data file generation.

Program : Q0PROB

Input : LCODEPWR.IN

Output : Q0PROB.OUT, [LCODEPWR.IN]

Function : Completes spectral analysis input data generation for Continuous GS codes which use a single augmenting bit per word by evaluating the probability of occurrence of each $q_0(x)$ sequence given the source stream dc level and code parameters specified in LCODEPWR.IN. Quotient probabilities will be written into Q0PROB.OUT and, if desired, also into the appropriate locations in LCODEPWR.IN.

Program : LCODEPWR

Input : LCODEPWR.IN

Output : LCODEPWR.OUT

Function : Evaluates the influence that line code rules have on the spectra of the transmitted signal through the technique outlined by Cariolaro and Tronca [24]. This is the only program which requires the presence of a co-processor for execution.

Miscellaneous Environment routines:

Program : LCESETUP

Input : none

Output : none

Function : Welcomes the user to the Line Coding Environment, accepts specification of a working directory, and creates or updates the LOG file in this directory. It is executed immediately upon entry to the Line Coding Environment when the environment is entered by running the LCE batch file.

Program :	NEWDIR
Input :	none
Output :	none
Function :	Changes the directory of operation and creates or updates the log file in the new directory.
Program :	TRANSFER
Input :	user specified
Output :	user specified
Function :	Transfers a bit stream from one file to another while optionally altering its word length and/or number of words.
Program :	UPDATE
Input :	none
Output :	LOG file
Function :	Appends to the log file information entered by the user.

The majority of the above programs utilize procedures which have been collected into libraries for convenient access. A brief description of the kind of routines contained in these libraries is given below.

Library :	BINLIB
Contents :	Catalogues global types, constants, and variables used in Line Coding Environment programs and contains routines which translate bit streams from character to binary representation and perform basic modulo-2 arithmetic.
Library :	IOLIB
Contents :	Contains a number of procedures which ease the pain in reading keyboard entries and writing to the screen.
Library :	FILELIB
Contents :	Contains procedures which expedite input and output to external fils and defines the structure in which bit stream history is retained in .BIT files.
Library :	SRCELIB
Contents :	Consists of routines used for random and sequential source stream generation.
Library :	SCRAMLIB
Contents :	Provides access to a routine which returns primitive irreducible polynomials usually used in scrambling and routines which perform reset and self-synchronizing scrambling and unscrambling.
Library :	7B8BLIB
Contents :	Contains routines which model the look-up tables for Alphabetic 7B8B encoding and decoding.

Library :	**PFLIB**
Contents :	**Contains procedures which encode and decode words according to the rules for Partially Flipped codes.**
Library :	**GSLIB**
Contents :	**Contains declarations for maximum GS parameter values and routines which implement Guided Scrambling encoding and decoding procedures.**
Library :	**LPRFLIB**
Contents :	**Defines the criteria by which line code performance is measured and provides routines to initialize and evaluate these metrics.**
Library :	**PWRLIB**
Contents :	**Contains subroutines which perform matrix operations an iteration required by LCODEPWR.**

APPENDIX 5

GLOSSARY

alphabetic codes

A form of line coding implemented with look-up tables. A set or sets of codewords are preselected, usually on the basis of low word disparity. Each set is called an "alphabet", the code "alphabetic". Each m-bit source word is assigned an n-bit codeword representation (n>m) from each set, leading to the mBnB notation with which alphabetic codes are commonly denoted.

balanced signal

A signal with a finite digital sum variation. In the long term, these signals contain an equal number of symbols of each symbol value. The continuous component of their power spectral density falls to zero as the frequency approaches zero, indicating a constant signal dc level.

block codes

Codes which translate m information bits to n encoded bits (n>m) on a word by word basis.

bit

A single digit in a binary signal. A contraction of *b*inary dig*it*.

bit stream

An unlimited sequence of bits.

codeword

A word in the encoded signal.

decoder

A device which attempts to restore the received, encoded signal to the original message.

digital sum variation (DSV)

Usually defined as the difference between the maximum and minimum running digital sum values in any coded sequence. Due to the unusual definition for RDS in this thesis and the desire to compare the DSV of codes developed in this thesis with those in the literature, digital sum variation is calculated as:

$$\mathrm{DSV} = {}^{1}/_{2} * (\mathrm{RDS}_{max} - \mathrm{RDS}_{min}) \ .$$

efficiency (of a line code)

A measure of the amount of information transmitted with respect to the number of symbols used to transmit this information. When source and encoded symbols take on the same number of levels, it is calculated by:

$$\text{code efficiency} = \frac{\text{\# of information symbols}}{\text{\# of transmitted symbols}} \ .$$

encoder

A device which transforms a source signal into an encoded signal for transmission.

error extension

The multiplication of errors which may occur during decoding.

frame

A reference indicating payload position within a symbol stream. Framing is the process of locating and retaining this reference.

Guided Scrambling (GS coding)

A new, balanced, non-alphabetic line code reported in this thesis developed specifically for high bit rate optical fiber transmission systems.

line coding

The modification of a source symbol sequence to allow for proper signal reception in the presence of transmission impairments, and subsequent restoration of the original signal.

Line Coding Environment

A collection of line code simulation and analysis routines which have been integrated into the Pull-Down Menu Shell. This environment runs on an IBM PC or compatible machine, and is available from the Alberta Telecommunications Research Centre.

majority bit

A bit whose value is most common within a word.

mark

A "1" in a binary signal.

message

The information to be conveyed through a communications network.

minority bit

A bit whose value is least common within a word.

Pull-Down Menu Shell

A user-friendly interface written for the IBM PC family of machines which allows for execution of programs through selection from user defined pull-down menus or from keyboard entry.

quotient selection set

The group of quotients from which the Guided Scrambling encoder selection mechanism must select a word for transmission.

receiver

A device which accepts and demodulates a transmitted signal.

running digital sum (RDS)

The accumulated sum of digit values in a symbol stream from some arbitrary time origin. Bit stream digits are usually assigned values of +0.5 for a mark, and -0.5 for a space. However, to simplify analysis in this thesis, digit values have been assigned as +1 for a mark, -1 for a space.

scrambling

An unbalanced line coding technique which does not require augmentation of the source bit stream but relies instead on statistical randomization. Reset and self-synchronizing scrambling are two established scrambling techniques.

source

The origin of a message signal.

space

A "0" in a binary signal.

stand-alone bit

A bit found between bits of opposing polarity such that transitions appear both preceding and following it.

transition density

The number of level transitions in a bit stream divided by the number of bits.

transmitter

A device which modulates a signal for transmission over a c[illegible]ions channel.

user

The destination of a message.

weight (of a polynomial)

The number of non-zero coefficients in a polynomial.

word (in a bit stream)

A specific, limited number of successive bits with defined starting and ending positions within a bit stream.

word disparity (WD)

The value of the running digital sum calculated over the period of one word.

word-end running digital sum (WRDS)

The running digital sum sampled at the end of a word within a bit stream.