JPL PUBLICATION 82-71

Reed-Solomon Encoders — Conventional vs Berlekamp's Architecture

Marvin Perlman
Jun-Ji Lee

December 1, 1982

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

CONTENTS

Tables

CONTENTS (contd)

Figures

NOTE

This report was originally a JPL interoffice memorandum (IOM No.: 3610-81-119 ISPM) entitled "Reed-Solomon Encoders — Conventional Versus Berlekamp's Architecture," dated July 10, 1981.

ABSTRACT

Concatenated coding has been adopted by the National Aeronautics and Space Administration of the United States of America for interplanetary space missions. NASA's Jet Propulsion Laboratory is employing concatenated coding with a convolutional inner code and a Reed-Solomon outer code for spacecraft telemetry.

This paper compares conventional RS encoders with those that incorporate two ingenious architectural features due to E. R. Berlekamp. Berlekamp's architecture approximately halves the number of multiplications of a set of fixed arguments by any RS codeword symbol. The fixed arguments and the RS symbols are taken from a nonbinary finite field. Each set of multiplications is bit-serially performed and completed during one (bit-serial) symbol shift. Berlekamp's architecture eliminates all firmware employed by conventional RS encoders.

I. BACKGROUND

Reed-Solomon (RS) codes are a special case of the nonbinary generalization of Bose-Chaudhuri-Hocquenghem (BCH) codes. They are among the Maximum Distance Separable (MDS) codes which realize the maximum minimum Hamming distance possible for a linear code (Refs. 1 and 2). The interest in RS codes was primarily theoretical until the concept of concatenated coding was formulated and first introduced in Ref. 3. Concatenated coding has been adopted by the U.S. National Aeronautics and Space Administration (NASA) for interplanetary space missions (see Fig. 1). The inner code is a convolutional code, whereas the outer code is an RS code. The application of concatenated coding to NASA's Jet Propulsion Laboratory (JPL) spacecraft telemetry with a convolutional inner code and an RS outer code was first proposed and analyzed in Ref. 4. This was followed by a contract study: "Concatenated RS/Viterbi Channel Coding for Advanced Planetary Missions: Analysis, Simulations and Tests." Reference 5 is the final report of that study. Reference 6 presents a discussion of the Viterbi decoder which serves as a maximum likelihood decoder of the inner convolutional code.

An investigation undertaken at JPL of alternative communication systems for downlinking imaging and general science data appears in Ref. 7. This resulted in the adoption of concatenated RS/convolutional coding for imaging data from the Voyager spacecraft as a backup beyond Saturn encounter. Imaging data from the Galileo spacecraft will also be subjected to RS/convolutional coding. This decision is a consequence of the foregoing and subsequent investigations as exemplified in Ref. 8.

Concatenated RS/Viterbi channel performance tests were made at JPL using simulation of ideal and nonideal receiver system models. The results of these tests led to the adoption of RS/convolution coding for the NASA spacecraft of the International Solar Polar Mission (ISPM) (see Ref. 9). The same coding has since been adopted for the European Space Agency (ESA) spacecraft for ISPM. Experimental results of RS/Viterbi channel coding on system performance and its impact on deep space transmission of imaging information appears in Ref. 10. When used as an outer code, protection is provided against errors emanating from the inner Viterbi decoder. Viterbi decoding errors tend to occur in bursts whereby relatively few RS symbols are affected. The expected burst length and the density of bit errors within a burst bear some relation to the channel's signal-to-noise ratio. A

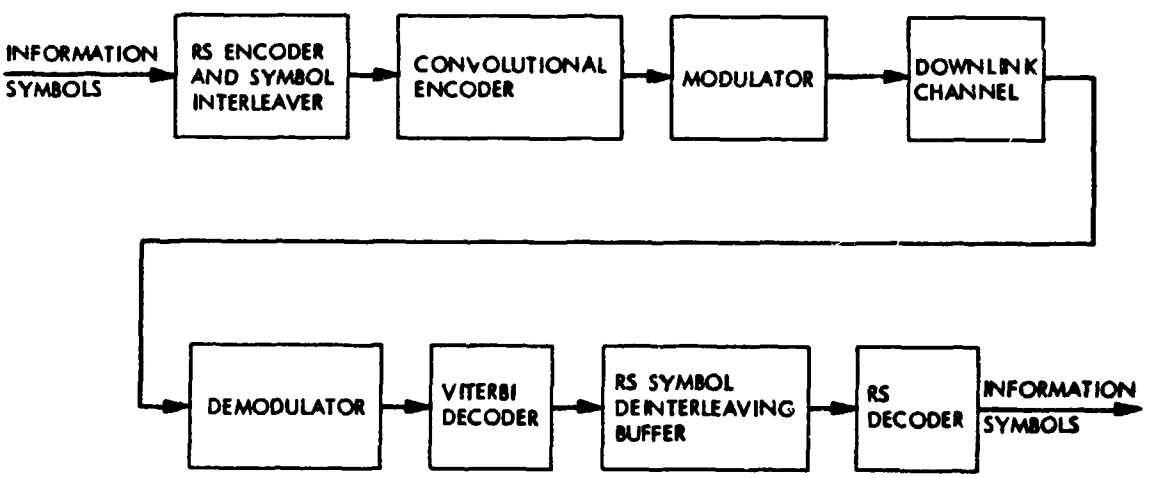

Figure 1. Concatenated Coding for a Spacecraft Telemetry Channel

performance analysis of the interleaved (255, 223) RS code in combating Viterbi decoding errors is given in Ref. 11.

II. PARAMETERS AND PROPERTIES OF RS CODES

The class of Reed-Solomon codes of interest for practical considerations has the following parameters:

J	the number of bits per symbol
$N = 2^J-1$	the total number of symbols per RS codeword
E	the symbol error correction capability
2E	the number of symbols representing checks
K = N-2E	the number of symbols representing information
I	the depth of symbol interleaving. That is, within a sequence of NI symbols comprising I RS codewords, consecutive symbols of a given RS codeword are separated by I-1 symbols belonging to other codewords

Note that J, E, and I are independent parameters.

The symbols of an (N,K) RS code are taken from a finite field of 2^J elements referred to as a Galois Field of order 2^J or simply $GF(2^J)$ (see Refs. 1 and 2). Every pair of distinct N-symbol codewords differs in at least 2E + 1 symbols. Thus an (N,K) RS code has a minimum Hamming distance of 2E +1 and is E symbol error-correcting. A received word with any combination of E or fewer symbols in error will be correctly decoded, whereas a received word containing more than E symbols in error will be incorrectly decoded with a probability of less than one chance in E factorial (i.e., E!).

Erroneous symbols of a received word confined to a region of E consecutive symbols or less are correctable. In terms of bits, a burst-error of length J(E-1) + 1 bits can affect at most E contiguous symbols. Hence all bursts of length J(E-1) + 1 bits or less are correctable. Symbol interleaving to a depth of I results in an (NI,KI) code which inherits its properties from the (N,K) RS code.

Each of the 2E symbols of an (N,K) RS codeword is a distinct linear combination of information symbols. Thus RS codes are linear. An (NI,KI) code is comprised of KI information symbols over which 2EI check symbols are computed. Every I^{th} symbol, starting with symbol 1,2,$\cdots$, or I, belongs to the same (N,K) RS codeword.

Symbol interleaving to a depth of I increases the burst-error correction capability to length J(EI-1) + 1 bits. If a received word of an (NI,KI) code contains a burst of length J(EI-1) + 1 bits or less, the number of erroneous symbols belonging to the same N-symbol word will not exceed E. Upon deinterleaving, each of the I N-symbol words will thus be correctly decoded.

Linearly combining any two codewords, not necessarily distinct, of a given (N,K) RS code results in a codeword. Each codeword may be viewed as a vector whose components (referred to as symbols) are field elements taken from $GF(2^J)$. Scalar multiplication and vector addition follow from the binary operations of "multiplication" and "addition" on the field elements. The foregoing is a restatement of the linearity of RS codes.

Every cyclic permutation of the symbols of an (N,K) RS codeword is a codeword. Thus RS codes are cyclic. Note that all cyclic codes are linear but the converse does not hold. Because of the cyclic property of an (N,K) RS code, it can be characterized by a generator polynomial g(x). The degree of g(x) is 2E, the number of check symbols. And g(x) has 2E distinct roots which are consecutive integer powers (excluding zero) of a primitive element in $GF(2^J)$.

Cyclic codes have a well-defined mathematical structure. Furthermore, encoders and decoders of cyclic codes are implementable by means of feedback shift registers (FSRs). However, unlike Bose-Chaudhuri-Hoquenghem (BCH) codes, RS codes are nonbinary. Thus each stage of the FSR must be capable of storing any one of 2^J J-bit symbols. Solid-state random-access memories (RAMs) are commonly used to serve as nonbinary FSR stages.

The Hamming weight enumerator for MDS codes (hence RS codes) is well known. "Separable" (in Maximum Distance Separable, MDS) and "systematic" are synonymous terms for codes whose information symbols occupy leading adjacent positions and are followed by check symbols. See Refs. 1, 2, and 6 for a detailed treatment of BCH and RS codes.

III. MATHEMATICAL CHARACTERIZATION OF RS CODES

Consider an (N,K) RS codeword

$$C = C_{N-1}C_{N-2} \cdots C_{2E+1}\ C_{2E} \cdots C_0, \text{ where } C_i \in GF(2^J) \qquad (1)$$

The polynomial

$$C(x) = C_{N-1}x^{N-1} + C_{N-2}x^{N-2} + \cdots + C_{2E}x^{2E} + C_{2E-1}x^{2E-1} + \cdots + C_0 \qquad (2)$$

over $GF(2^J)$ is termed a codeword polynomial. Every codeword polynomial contains

$$g(x) = \prod_{j=b}^{b+2E-1} (x - \gamma^j) = \sum_{i=0}^{2E} G_i x^i \qquad (3)$$

the generator polynomial of the code as a factor. Note that γ is any primitive element in $GF(2^J)$ and 2E consecutive powers (excluding zero) of γ (i.e., γ^b, γ^{b+1}, ..., γ^{b+2E-1}) are roots of g(x).

Encoding is the process of computing 2E check symbols over K information symbols such that the N (i.e., K+2E) symbols are coefficients of C(x) in (2) containing g(x) in (3) as a factor. Given the information polynomial

$$I(x) = C_{N-1}x^{K-1} + C_{N-2}x^{K-2} + \cdots + C_{2E} \qquad (4)$$

Check symbols C_{2E-1}, C_{2E-2}, ..., C_0 are computed as follows:

$$\frac{x^{2E}I(x)}{g(x)} = H(x) + \frac{r(x)}{g(x)}$$

$$x^{2E}I(x) = g(x)H(x) + r(x)$$

where

$$r(x) = C_{2E-1}x^{2E-1} + C_{2E-2}x^{2E-2} + \cdots + C_0 \qquad (5)$$

$$x^{2E}I(x) \equiv r(x) \bmod g(x) \qquad (6)$$

and

$$C(x) = x^{2E}I(x) + r(x) \equiv 0 \bmod g(x) \tag{7}$$

where $a(x) \equiv b(x) \bmod m(x)$ implies that $m(x)$ divides $a(x)-b(x)$, where $a(x)$ and $b(x)$ are polynomials over a field. Similarly, for integers $a \equiv b \bmod m$ implies that m divides $a-b$. The symbol "+" denotes sum modulo 2 (i.e., the exclusive-OR operation) and

$$-1 \equiv 1 \bmod 2$$

The polynomials $x^{2E}I(x)$ and $r(x)$ in (6) and (7) are nonoverlapping and the coefficients of $C(x)$ in (7) as explicitly shown in (2) represent an (N,K) RS codeword. Furthermore, $C(x)$ contains $g(x)$ as a factor.

IV. HARDWARE CONSIDERATIONS IN THE DESIGN OF RS ENCODERS

A. CONVENTIONAL ARCHITECTURE

A functional logic diagram of a conventional (N,K) RS encoder appears in Fig. 2. Assume the register (composed of 2E J-bit storage elements) of the FSR is initially cleared. With switches A and B in the up position, information symbols (i.e., coefficients of $I(x)$ in (4)) are sequentially entered and simultaneously delivered to the channel. Symbol C_{N-1} is entered first and C_{2E} last. Upon the entry of C_{2E}, the check symbols which are coefficients of $r(x)$ in (5) reside in the register where C_i is stored in x^i. At this time, switches A and B are placed into the down position. The check symbols, starting with C_{2E-1}, are then delivered to the channel while the register is cleared in preparation for the next set of K information symbols.

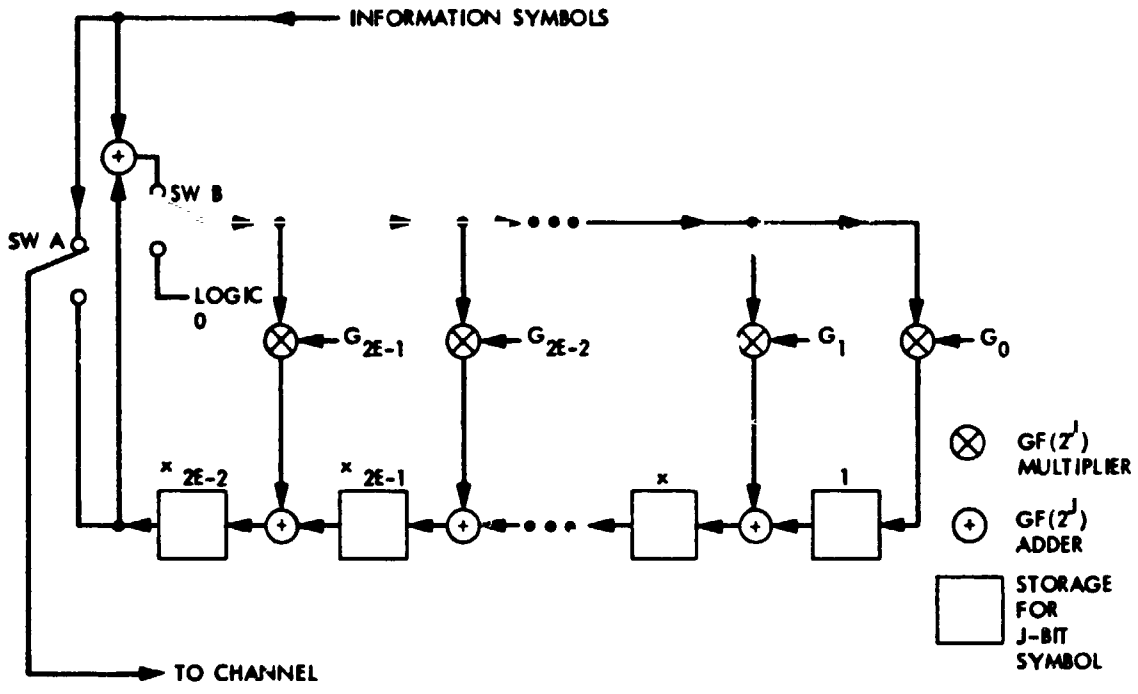

Figure 2. A Conventional (N,K) RS Encoder

The FSR in Fig. 2 accepts I(x) and computes r(x) by multiplying I(x) by x^{2E} and reducing the result modulo g(x) as given in (6). From (3), where G_{2E} necessarily equals 1,

$$x^{2E} \equiv G_{2E-1}x^{2E-1} + G_{2E-2}x^{2E-2} + \cdots + G_1 x + G_0 \bmod g(x)$$

Each of the 2E components

$$G_{2E-1},\ G_{2E-2},\ \cdots,\ G_1,\ G_0$$

is multiplied by the symbol appearing on the feedback path. The resulting 2E component vector is effectively added to the symbol string stored in the register after a symbol shift to the left has occurred. The incoming information symbols, C_{2E-1}, C_{2E-2}, $\cdots$, C_0, and the intermediately stored symbols are all members of $GF(2^J)$.

Consider a (255, 223) RS code where the field element α is a root of the primitive 8^{th} degree polynomial over GF(2)

$$f(x) = x^8 + x^7 + x^2 + x + 1 \tag{8}$$

Each nonzero element is expressible as an integer power of α, a generator of $GF(2^8)$. Since

$$\alpha^8 = \alpha^7 + \alpha^2 + \alpha + 1$$

every element is representable as a polynomial in α over GF(2) of degree less than 8. Thus

$$\alpha^n = u_7\,\alpha^7 + u_6\,\alpha^6 + \cdots + u_0 \tag{9}$$

where $u_i = 0$ or 1 and $0 \le n < 255$. The zero element (i.e., $00\cdots0$) corresponds to the constant 0 polynomial and is denoted by α^*.

A tabulation of a portion of $GF(2^8)$ generated by α appears as follows:

n of α^n	α^7	α^6	α^5	α^4	α^3	α^2	α	1
*	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0
2	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	1	0	0	0	0	0
6	0	1	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0
8	1	0	0	0	0	1	1	1
9	1	0	0	0	1	0	0	1
.					.			
.					.			
.					.			
24	0	1	1	1	0	0	0	1
25	1	1	1	0	0	0	1	0
26	0	1	0	0	0	0	1	1
27	1	0	0	0	0	1	1	0
28	1	0	0	0	1	0	1	1
29	1	0	0	1	0	0	0	1
30	1	0	1	0	0	1	0	1
31	1	1	0	0	1	1	0	1
32	0	0	0	1	1	1	0	1
33	0	0	1	1	1	0	1	0
.					.			
.					.			
.					.			
251	1	1	1	0	1	0	1	1
252	0	1	0	1	0	0	0	1
253	1	0	1	0	0	0	1	0
254	1	1	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1

The binary operation of "addition" defined on the field elements is termwise sum modulo 2 (i.e., vector addition over GF(2)).

Example 1

$$\begin{array}{rcl} & 1\,0\,0\,0\,0\,1\,1\,1 & (\alpha^{8}) \\ + & 1\,0\,0\,0\,0\,1\,1\,0 & (\alpha^{27}) \\ \hline & 0\,0\,0\,0\,0\,0\,0\,1 & (\alpha^{0}) \end{array}$$

□

Addition of RS symbols is readily implementable with 2-input Exclusive-OR gates.

The binary operation of "multiplication" defined on the field elements is

$$(u_7\,\alpha^7 + u_6\,\alpha^6 + \cdots + u_0)\,(v_7\,\alpha^7 + v_6\,\alpha^6 + \cdots + v_0)$$

with the result reduced modulo

$$f(\alpha) = \alpha^8 + \alpha^7 + \alpha^2 + \alpha + 1$$

The coefficients are members of GF(2) and subject to the rules of modulo 2 arithmetic.

Each multiplier of an RS encoder has one argument fixed, namely G_i, a coefficient of g(x). A hardware multiplier of an arbitrary field element by a fixed field element is given in Ref. 1 (chapter 2). Such a multiplier would be required for each distinct nonzero G_i which does not equal α^0 (i.e., 00···01).

Another method follows from the property

$$\alpha^i\,\alpha^j = \alpha^{(i+j)} \bmod 255$$

Example 2

$$\alpha^9 \quad (1\ 0\ 0\ 0\ 1\ 0\ 0\ 1)$$

$$\alpha^{24} \quad (0\ 1\ 1\ 1\ 0\ 0\ 0\ 1)$$

$$\alpha^{24}\,\alpha^9 = \alpha^{33} \quad (0\ 0\ 1\ 1\ 1\ 0\ 1\ 0) \qquad \square$$

The conventional approach for multiplying two field elements employs two read-only-memories (ROMs). The addresses of one ROM correspond to the field elements ($u_7\ u_6 \cdots u_0$ in GF (2^8)), and the content of each address is the binary representation of the log to the base α of the corresponding field element. The addresses of the other ROM correspond to the logs expressed in binary, and the content of each address is the antilog of the corresponding log. Multiplication in GF(2^8) utilizing the tables of logs and antilogs may be realized as follows.

(1) The logs of each of two field elements are sequentially read and stored.

(2) The 8-bit binary representations of the logs are added (as positional binary numbers) modulo 255. An overflow bit (2^8) is treated as an end-around carry resulting in casting out 2^8-1.

(3) The antilog corresponding to log of the product (derived in step 2) is then read out.

Example 3

	$\log_\alpha$	
$u_7u_6u_5u_4u_3u_2u_1u_0$	$b_7b_6b_5b_4b_3b_2b_1b_0$	
1 0 0 1 0 0 0 1	0 0 0 1 1 1 0 1	(29)
0 1 0 1 0 0 0 1	1 1 1 1 1 1 0 0	(252)
	0 0 0 1 1 0 0 1	
	→ 1	
	0 0 0 1 1 0 1 0	(26)

	$\log_\alpha$	antilog_α
	$b_7b_6b_5b_4b_3b_2b_1b_0$	$u_7u_6u_5u_4u_3u_2u_1u_0$
(26)	0 0 0 1 1 0 1 0	0 1 0 0 0 0 1 1

□

Note that $00\cdots0$ and $11\cdots1$ (255_{10}) have the same antilog.

If a fixed operand is 1 (α^0), multiplication is the identity operation realizable with J wires. If either operand is 0 (α^*), multiplication by 0 is implementable by logic external to the ROMs.

The most economical method (in terms of memory requirements) of interleaving a depth of I is to replace each of the 2E nonbinary stages with I stages. The 2EI stage FSR is described by the generator polynomial

$$\hat{g}(x) = \prod_{j=b}^{b+2E-1} (x^I - \alpha^j) = \sum_{i=0}^{2E} G_i (x^I)^i \qquad (10)$$

where the indeterminate x in (2) is replaced with x^I. The $\hat{g}(x)$ in (10) characterizes an (NI,KI) RS code where every I^{th} symbol starting with symbol 1, 2, $\cdots$, or I belongs to an (N,K) RS codeword characterized by g(x) in (3). Clearly the G_i's associated with g(x) and $\hat{g}(x)$ are identical.

In a conventionally designed RS encoder a single ROM (containing both tables) and binary adder (with end-around carry) could be sequentially shared by each multiplier having a different G_i (where $G_i \neq \alpha^0$) as one of its operands. The cost

of this reduced complexity is the increase in time needed for K sets of multiplications per (N,K) RS codeword. A set could contain up to 2E multiplications associated with 2E distinct G_i's none of which is equal to α^0.

B. BERLEKAMP'S ARCHITECTURE

The RS encoder design due to Berlekamp (Ref. 12) incorporates two ingenious features. First, the number of multiplications per symbol shift is approximately halved by selecting a g(x) whose 2E roots are E reciprocal pairs. That is, in (3),

$$\alpha^{b+i}\,\alpha^{(b+2E-1)\,-i} = \alpha^N = 1 \quad 0 \le i \le E \tag{11}$$

In expanded form, g(x) is a self-reciprocal polynomial where (over the range of i in (11))

$$G_{2E-i} = G_i \text{ and } G_{2E} = G_0 = 1\ (00\cdots01)$$

Second, and more significant, Berlekamp formulated a hardware design of bit-serial multipliers over $GF(2^J)$ which is compatible with the serial organization of RS encoders. One operand is any of the 2^J field elements. The other is a vector whose components are fixed distinct G_i's representing coefficients of g(x).

In the design of an (N,K) RS encoder, two parameters affected the complexity of the circuitry associated with multiplication. These parameters are discussed in connection with a (255, 223) RS encoder unless stated otherwise.

The elements of $GF(2^8)$ form a vector space of dimension 8. One parameter is β where

$$1,\ \beta,\ \beta^2,\ \cdots,\ \beta^7$$

is a basis, a set of linearly independent vectors which spans the vector space of $GF(2^8)$. In the case where β equals α, a generator (associated with (8) and (9)), results in the basis made up of the unit vectors α^0 $(00\cdots001)$, α^1 $(00\cdots010)$, etc. Any element in $GF(2^8)$ that is <u>not</u> a member of a <u>subfield</u> may serve as β Since,

$$GF(2) \subset GF(2^2) \subset GF(2^4) \subset GF(2^8)$$

β cannot be selected from the 16 elements in $GF(2^4)$. Each element of $GF(2^8)$ is a root of

$$x^{2^8} - x = x(x^{255} - 1) = 0$$

Each element of $GF(2^4)$ is also a root of

$$x^{2^4} - x = x(x^{15} - 1) = 0$$

Let $\{\alpha^y\}$ be the set of 15 nonzero roots of unity. Then

$$(\alpha^y)^{15 \bmod 255} = 1 = \alpha^0$$

and

$$\begin{aligned} 15y &\equiv 0 \bmod 255 \\ y &\equiv 0 \bmod \frac{255}{(15,255)} \\ y &\equiv 0 \bmod 17 \end{aligned}$$

(where (r,s) denotes the greatest common divisor of r and s). Thus 0 (α^*) and

$$\alpha^{17k} \text{ for } 0 \le k < 15$$

compose the subfield $GF(2^4)$, a vector space of dimension 4. Thus there can be at most 4 linearly independent vectors in the set

$$1, \alpha^{17k}, \alpha^{(17kx2) \bmod 255}, \cdots, \alpha^{(17kx7) \bmod 255}$$

where $k = 0, 1, \cdots,$ or 14.

For each basis

$$\{1, \beta, \beta^2, \cdots, \beta^7\} = \{\beta^i\}$$

in $GF(2^8)$, a dual basis (also called a complementary and a trace-orthogonal basis) is determined (see Refs. 1, 2, and 13). The concept of a trace of a finite field element is involved in the development of a dual basis.

Consider $GF(p^n)$, a finite field of p^n elements over GF(p) where p is a prime. The trace Tr is a function on $GF(p^n)$ defined by

$$Tr(\gamma) = \sum_{i=0}^{n-1} \gamma^{p^i} \quad \text{where } \gamma \in GF(p^n)$$

The trace has the following properties:

(1) $Tr(\gamma) \in GF(p)$

(2) $Tr(\gamma+\delta) = Tr(\gamma) + Tr(\delta)$

(3) $Tr(c\gamma) = cTr(\gamma)$ where $c \in GF(p)$

A proof for each follows:

(1)
$$[Tr(\gamma)]^p = \left(\gamma + \gamma^p + \gamma^{p^2} + \cdots + \gamma^{p^{n-1}}\right)^p$$
$$= \gamma^p + \gamma^{p^2} + \gamma^{p^3} + \cdots + \gamma^{p^n}$$
$$= Tr(\gamma) \text{ since } \gamma^{p^n} = \gamma$$

Thus $[Tr(\gamma)]^p = Tr(\gamma)$ implies that $Tr(\gamma) \in GF(p)$.

(2)
$$Tr(\gamma+\delta) = \sum_{i=0}^{n-1} (\gamma+\delta)^{p^i} = \sum_{i=0}^{n-1} (\gamma^{p^i} + \delta^{p^i})$$

$$= \sum_{i=0}^{n-1} \gamma^{p^i} + \sum_{i=0}^{n-1} \delta^{p^i} = Tr(\gamma) + Tr(\delta)$$

(3)
$$Tr(c\gamma) = \sum_{i=0}^{n-1} (c\gamma)^{p^i} = \sum_{i=0}^{n-1} c^{p^i} \gamma^{p^i}$$

$$c^{p^i} = (\cdots (c^p)^p \cdots)^p = c \text{ for } i > 1$$

since $c^p = c$.

$$Tr(c\gamma) = \sum_{i=0}^{n-1} c\gamma^{p^i} = c \sum_{i=0}^{n-1} \gamma^{p^i} = cTr(\gamma)$$

Example 4

Given $GF(2^4)$ generated by α, a root of the primitive polynomial $x^4 + x + 1$ over GF(2). The trace of each of 16 elements is tabulated as follows:

n of α^n	$\alpha^3\alpha^2\alpha$ 1	$Tr(\alpha^n)$
*	0 0 0 0	0
0	0 0 0 1	0
1	0 0 1 0	0
2	0 1 0 0	0
3	1 0 0 0	1
4	0 0 1 1	0
5	0 1 1 0	0
6	1 1 0 0	1
7	1 0 1 1	1
8	0 1 0 1	0
9	1 0 1 0	1
10	0 1 1 1	0
11	1 1 1 0	1
12	1 1 1 1	1
13	1 1 0 1	1
14	1 0 0 1	1

From the definition of the trace

$$Tr(\alpha) = \alpha + \alpha^2 + \alpha^4 + \alpha^8 = 0000 = 0$$

$$Tr(\alpha^3) = \alpha^3 + \alpha^6 + \alpha^{12} + \alpha^{24} \quad (\alpha^{24 \bmod 15} = \alpha^9)$$

$$= \alpha^3 + \alpha^6 + \alpha^{12} + \alpha^9 = 0001 = 1$$

$$Tr(\alpha^*) = Tr(0) = 0 + 0 + 0 + 0 = 0$$

$$Tr(\alpha^0) = Tr(1) = 1 + 1 + 1 + 1 = 0$$

□

$$\begin{aligned} Tr(\alpha^5) &= \alpha^5 + \alpha^{10} + \alpha^{20} + \alpha^{40} \\ &= \alpha^5 + \alpha^{10} + \alpha^5 + \alpha^{10} = 0 \\ Tr(\alpha^7) &= \alpha^7 + \alpha^{14} + \alpha^{28} + \alpha^{56} \\ &= \alpha^7 + \alpha^{14} + \alpha^{13} + \alpha^{11} = 0001 = 1 \end{aligned}$$ □

From the linear property, the trace of α^n is

$$\begin{aligned} Tr(\alpha^n) &= Tr(u_3\alpha^3 + u_2\alpha^2 + u_1\alpha + u_0) \\ &= u_3Tr(\alpha^3) + u_2Tr(\alpha^2) + u_1Tr(\alpha) + u_0Tr(\alpha^0) \\ &= u_3 \end{aligned}$$

since $Tr(\alpha^3) = 1$ and $Tr(\alpha^2) = Tr(\alpha) = Tr(\alpha^0) = 0$ in $GF(2^4)$ in example 4.

In $GF(2^8)$ generated by α, a root of Eq. (8), the trace of an element as represented by α^n in (9) is

$$Tr(\alpha^n) = u_7 + u_6 + u_5 + u_4 + u_3 + u_2 + u_1$$

since

$$Tr(\alpha^i) = 1 \text{ for } 1 \leq i \leq 7 \text{ and } Tr(\alpha^0) = 0$$

For each basis $\{\beta^i\}$ in $GF(2^8)$, α^n is also representable as

$$v_0\ell_0 + v_1\ell_1 + \cdots + v_7\ell_7 \tag{12}$$

where

$$v_i = Tr(\beta^i\alpha^n)$$

The set

$$\{\ell_0, \ell_1, \cdots, \ell_7\} = \{\ell_j\}$$

is a basis dual to the basis $\{\beta^i\}$ such that

$$Tr(\beta^i \ell_j) = \begin{cases} 1 \text{ for } 0 \leq i = j < 8 \\ 0 \text{ for } 0 \leq i \neq j < 8 \end{cases} \tag{13}$$

Given an element α^n in $GF(2^8)$. Its components in the dual basis are readily computed as follows.

$$\alpha^n \leftrightarrow \sum_{j=0}^{7} v_j \ell_j$$

Thus,

$$\beta^i \alpha^n \leftrightarrow \sum_{j=0}^{7} = v_j \beta^i \ell_j$$

and

$$Tr(\beta^i \alpha^n) = \sum_{j=0}^{7} v_j \; Tr(\beta^i \ell_j) = v_i \tag{14}$$

from property (3) of a trace and (13).

A selection of a basis $\{\beta^i\}$ and a determination of its dual basis $\{\ell_j\}$ are illustrated in example 5.

Example 5

Given $GF(2^6)$ generated by α, a root of the primitive 6^{th} degree polynomial $x^6 + x^5 + x^2 + x + 1$ over GF(2). Contained within $GF(2^6)$ are the subfields $GF(2^2)$ and $GF(2^3)$, and the subfield GF(2) is contained in both $GF(2^2)$ and $GF(2^3)$.

Each element of $GF(2^2)$ is a root of

$$x^{2^2} - x = x(x^3-1) = 0$$

Let $\{\alpha^w\}$ be the set of three nonzero roots of unity. Then

$$(\alpha^w)^{3 \bmod 63} = 1 = \alpha^0$$

and

$$3w \equiv 0 \bmod 63$$

$$w \equiv 0 \bmod \frac{63}{(3,63)}$$

$$w \equiv 0 \bmod 21$$

Thus the elements in $GF(2^6)$ which compose the subfield $GF(2^2)$ are

$$\alpha^*, \ \alpha^0, \ \alpha^{21}, \ \alpha^{42}$$

Each element in $GF(2^3)$ is a root of

$$x^{2^3} - x = x(x^7-1) = 0.$$

Let $\{\alpha^y\}$ be the set of 7 nonzero roots of unity. Then

$$(\alpha^y)^{7 \bmod 63} = 1 = \alpha^0$$

and

$$7y \equiv 0 \bmod 63$$

$$y \equiv 0 \bmod \frac{63}{(7,63)}$$

$$y \equiv 0 \bmod 9$$

Thus the elements in $GF(2^6)$ which compose the subfield $GF(2^3)$ are

$$\alpha^*, \ \alpha^0, \ \alpha^9, \ \alpha^{18}, \ \alpha^{27}, \ \alpha^{36}, \ \alpha^{45}, \ \alpha^{54}$$

Any element not contained in $GF(2^2)$ and not contained in $GF(2^3)$ may serve as β in forming the basis

$$\{1, \beta, \beta^2, \cdots, \beta^5\} = \{\beta^i\}$$

in $GF(2^6)$. In this example β equal to α^3 was selected. In Table 1 each field element, α^n, in $GF(2^6)$ is represented in two ways. Namely,

$$\alpha^n = u_5\alpha^5 + u_4\alpha^4, + \cdots + u_0$$

where

$$\alpha^6 = \alpha^5 + \alpha^2 + \alpha + 1$$

and

$$v_0\ell_0 + v_1\ell_1 + \cdots + v_5\ell_5 \leftrightarrow \alpha^n$$

where

$$v_i = \mathrm{Tr}(\beta^i\alpha^n) = \mathrm{Tr}(\alpha^{n+3i})$$

and

$$\mathrm{Tr}(\alpha^n) = u_5 + u_4 + u_3 + u_2 + u_1$$

The basis $\{\beta^i\}$ in $GF(2^6)$ is

$$\{1, \beta, \beta^2, \beta^3, \beta^4, \beta^5\} = \{1, \alpha^3, \alpha^6, \alpha^9, \alpha^{12}, \alpha^{15}\}$$

The entries in column ℓ_0 corresponding to α^n are

$$v_0 = \mathrm{Tr}(\alpha^n)$$

Table 1. Representations of Elements in $GF(2^6)$

n of α^n	$\alpha^5\alpha^4\alpha^3\alpha^2\alpha$ 1	$Tr(\alpha^n)$	$\ell_0\ell_1\ell_2\ell_3\ell_4\ell_5$	
*	0 0 0 0 0 0	0	0 0 0 0 0 0	
0	0 0 0 0 0 1	0	0 1 1 0 1 0	
1	0 0 0 0 1 0	1	1 1 0 1 0 1	
2	0 0 0 1 0 0	1	1 1 1 1 0 1	
3	0 0 1 0 0 0	1	1 1 0 1 0 0	
4	0 1 0 0 0 0	1	1 0 1 0 1 0	
5	1 0 0 0 0 0	1	1 1 1 0 1 1	
6	1 0 0 1 1 1	1	1 0 1 0 0 1	
7	1 0 1 0 0 1	0	0 1 0 1 0 1	
8	1 1 0 1 0 1	1	1 1 0 1 1 0	
9	0 0 1 1 0 1	0	0 1 0 0 1 1	
10	0 1 1 0 1 0	1	1 0 1 0 1 1	
11	1 1 0 1 0 0	1	1 0 1 1 0 0	
12	0 0 1 1 1 1	1	1 0 0 1 1 0	
13	0 1 1 1 1 0	0	0 1 0 1 1 0	
14	1 1 1 1 0 0	0	0 1 1 0 0 0	
15	0 1 1 1 1 1	0	0 0 1 1 0 0	
16	1 1 1 1 1 0	1	1 0 1 1 0 1	
17	0 1 1 0 1 1	1	1 1 0 0 0 1	
18	1 1 0 1 1 0	0	0 1 1 0 0 1	
19	0 0 1 0 1 1	0	0 1 1 0 1 1	
20	0 1 0 1 1 0	1	1 0 0 0 1 0	
21	1 0 1 1 0 0	1	1 1 0 0 1 0	
22	1 1 1 1 1 1	1	1 1 0 1 1 1	
23	0 1 1 0 0 1	0	0 0 0 1 0 0	$\ell_3 = \alpha^{23}$
24	1 1 0 0 1 0	1	1 0 0 1 0 0	
25	0 0 0 0 1 1	1	1 0 1 1 1 1	
26	0 0 0 1 1 0	0	0 0 1 0 0 0	$\ell_2 = \alpha^{26}$
27	0 0 1 1 0 0	0	0 0 1 0 0 1	
28	0 1 1 0 0 0	0	0 1 1 1 1 0	
29	1 1 0 0 0 0	0	0 1 0 0 0 1	
30	0 0 0 1 1 1	0	0 1 0 0 1 0	
31	0 0 1 1 1 0	1	1 1 1 1 0 0	
32	0 1 1 1 0 0	1	1 0 0 0 1 1	
33	1 1 1 0 0 0	1	1 0 0 1 0 1	
34	0 1 0 1 1 1	1	1 1 1 0 0 0	
35	1 0 1 1 1 0	0	0 0 0 1 1 1	
36	1 1 1 0 1 1	0	0 0 1 0 1 0	
37	0 1 0 0 0 1	1	1 1 0 0 0 0	
38	1 0 0 0 1 0	0	0 0 1 1 1 0	
39	1 0 0 0 1 1	0	0 1 0 1 0 0	

Table 1. Representations of Elements in GF(2^6) (contd)

n of α^n	$\alpha^5\alpha^4\alpha^3\alpha^2\alpha\ 1$	$Tr(\alpha^n)$	$\ell_0\ell_1\ell_2\ell_3\ell_4\ell_5$	
40	1 0 0 0 0 1	1	1 0 0 0 0 1	
41	1 0 0 1 0 1	0	0 1 1 1 0 0	
42	1 0 1 1 0 1	1	1 0 1 0 0 0	
43	1 1 1 1 0 1	0	0 0 0 0 1 0	$\ell_4 = \alpha^{43}$
44	0 1 1 1 0 1	1	1 1 1 0 0 1	
45	1 1 1 0 1 0	0	0 1 0 0 0 0	$\ell_1 = \alpha^{45}$
46	0 1 0 0 1 1	0	0 0 0 1 0 0	
47	1 0 0 1 1 0	1	1 1 0 0 1 1	
48	1 0 1 0 1 1	1	1 0 0 0 0 0	$\ell_0 = \alpha^{48}$
49	1 1 0 0 0 1	0	0 0 1 0 1 1	
50	0 0 0 1 0 1	1	1 0 0 1 1 1	
51	0 0 1 0 1 0	0	0 0 0 0 0 1	$\ell_5 = \alpha^{51}$
52	0 1 0 1 0 0	0	0 1 0 1 1 1	
53	1 0 1 0 0 0	0	0 0 1 1 1 1	
54	1 1 0 1 1 1	0	0 0 0 0 1 1	
55	0 0 1 0 0 1	1	1 0 1 1 1 0	
56	0 1 0 0 1 0	0	0 1 1 1 1 1	
57	1 0 0 1 0 0	0	0 0 0 1 1 0	
58	1 0 1 1 1 1	0	0 1 1 1 0 1	
59	1 1 1 0 0 1	1	1 1 1 1 1 1	
60	0 1 0 1 0 1	0	0 0 1 1 0 1	
61	1 0 1 0 1 0	1	1 1 1 0 1 0	
62	1 1 0 0 1 1	1	1 1 1 1 1 0	

$\beta = \alpha^3$

	ℓ_0	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5
1	α^{48}	α^{45}	α^{26}	α^{23}	α^{43}	α^{51}
β	α^{51}	α^{48}	α^{29}	α^{26}	α^{46}	α^{54}
β^2	α^{54}	α^{51}	α^{32}	α^{29}	α^{49}	α^{57}
β^3	α^{57}	α^{54}	α^{35}	α^{32}	α^{52}	α^{60}
β^4	α^{60}	α^{57}	α^{38}	α^{35}	α^{55}	α^{0}
β^5	α^{0}	α^{60}	α^{41}	α^{38}	α^{58}	α^{3}

$\beta^i\ell_j$

	ℓ_0	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5
1	1	0	0	0	0	0
β	0	1	0	0	0	0
β^2	0	0	1	0	0	0
β^3	0	0	0	1	0	0
β^4	0	0	0	0	1	0
β^5	0	0	0	0	0	1

$Tr(\beta^i\ell_j)$

whereas the entries in column ℓ_1 are

$$v_1 = \mathrm{Tr}(\beta\alpha^n) = \mathrm{Tr}(\alpha^{n+3})$$

which is $\mathrm{Tr}(\alpha^n)$ cyclically shifted upward three places excluding $\mathrm{Tr}(\alpha^*)$. The remaining columns are similarly formed. The dual basis $\{\ell_j\}$ is

$$\{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4, \ell_5\} = \{\alpha^{48}, \alpha^{45}, \alpha^{26}, \alpha^{23}, \alpha^{43}, \alpha^{51}\}$$

The elements $\beta^i\ell_j \in GF(2^6$ and $\mathrm{Tr}(\beta^i\ell_j) \in GF(2)$, respectively, are entries in the foregoing tables.

As previously asserted,

$$\mathrm{Tr}(\beta^i\ell_j) = \begin{cases} 1 \text{ for } 0 \le i = j < 6 \\ 0 \text{ for } 0 \le i \ne j < 6 \end{cases} \qquad \square$$

The trace $\mathrm{Tr}(\beta^i\ell_j)$ plays a role in determining the components of a field element α^n in the dual basis $\{\ell_j\}$ as shown in (14). The product of an arbitrary field element with a fixed coefficient of g(x) is realized bit-serially in the dual basis.

The order of a nonzero element α^k in $GF(2^8)$ is

$$\frac{255}{(k, 255)}$$

If (k, 255) is 1, α^k is of order 255, hence primitive. There are a total of $\phi(255)$ or 128 primitive elements in $GF(2^8)$, where $\phi(n)$ is the number of integers no greater than n that are relatively prime to n. (An integer i and n are relatively prime if (i,n) is 1.)

Let γ be a primitive element in $GF(2^8)$. Corresponding to γ is a generator polynomial

$$g(x) = \prod_{j=b}^{b+2E-1} (x-\gamma^j) = \sum_{i=0}^{2E} G_i x^i$$

for a (255, 255-2E) RS code.

From (11)

$$2b + 2E - 1 = 255$$

For an E of 16,

$$b = 112 \text{ and } b + 2E - 1 = 143$$

and given a primitive element γ in $GF(2^8)$,

$$g(x) = \prod_{j=112}^{143} (x-\gamma^j) = \sum_{i=0}^{32} G_i x^i \qquad (15)$$

is a self-reciprocal generator polynomial for a (255, 223) RS code.

Given that γ equal to α^k is primitive, the expanded g(x) in (15) will be the same for the reciprocal of γ (i.e., γ^{-1} equal to α^{255-k}). Thus there are 64 distinct self-reciprocal polynomials over $GF(2^8)$ that could serve as the codes' generator polynomial g(x). For an (N,K) RS code, there are $\phi(N)/2$ distinct self-reciprocal polynomials over $GF(2^J)$ from which g(x) may be selected.

The field element β used to form a basis in $GF(2^8)$ and the field element γ in (14) govern the complexity of the bit-serial hardware multiplier in the Berlekamp RS encoder architecture. Element β can be selected from among 240 elements in $GF(2^8)$ — i.e., 256 less the 16 elements comprising the subfield $GF(2^4)$. Element γ can be selected from among 64 pairs of reciprocal primitive elements in $GF(2^8)$ independently of the choice of β.

For a given basis $\{\beta^i\}$ in $GF(2^8)$, its dual basis $\{\ell_j\}$ is determined as illustrated in Example 5 for a field of lower order. Corresponding to a given primitive element γ (or γ^{-1}) in $GF(2^8)$, the coefficients G_i of $g(x)$ in the expanded form in (15) are determined where

$$G_0 = G_{32} = 1 \text{ and } G_{32-i} = \qquad 1 \leq i \leq E$$

The 16 coefficients

$$G_1, G_2, \cdots, G_{16}$$

represent a largest set of distinct coefficients not equal to 1 (α^0).

Bit-serial multiplication of the vector

$$G_0, G_1, G_2, \cdots, G_{16}$$

by a field element z (i.e., an RS symbol) is realized as follows:

A linear binary matrix (i.e., an array of Exclusive-OR gates) is used to compute

$$T_\ell(z) = Tr(z \cdot G_\ell)$$

Since

$$z \cdot G_\ell = \sum_{j=0}^{7} z_j^{(\ell)} \ell_j$$

in the dual basis,

$$\mathrm{Tr}[\beta^i(z \cdot G_\ell)] = \sum_{j=0}^{7} z_j^{(\ell)} \mathrm{Tr}(\beta^i \ell_j)$$

$$= T_\ell(\beta^i z) = z_i^{(\ell)} \tag{16}$$

The simultaneous application of $T_0, T_1, \cdots, T_{16}$ to a stored z yields

$$\{z_0^{(\ell)}\} \quad \text{for} \quad 0 \leq \ell \leq 16$$

from (16) where i = 0. Note that $\{z_0^{(\ell)}\}$ is the first component of the products $zG_0, zG_1, \cdots, zG_{16}$. Subsequently, z is replaced by βz and a simultaneous application of $T_0, T_1, \cdots, T_{16}$ to a stored βz yields

$$\{z_1^{(\ell)}\} \text{ for } 0 \leq \ell \leq 16$$

(from (16) where i = 1), the second components of $\{zG_\ell\}$. Similarly, replacing βz by β(βz) and applying $\{T_\ell\}$ yields the third component of $\{zG_\ell\}$ and so on. It will be shown that βz is simply deri.ed from z.

The form of the functions $\{T_\ell\}$ is

$$T_\ell(z) = \mathrm{Tr}(zG_\ell) = \sum_{j=0}^{7} z_j \mathrm{Tr}(\ell_j G_\ell) \tag{17}$$

For every z, the output of T_ℓ is the modulo 2 sum (i.e., Exclusive-OR) of those components z_j's in the dual basis for which

$$\mathrm{Tr}(\ell_j G_\ell) = 1$$

A functional logic diagram of an (N,K) RS encoder utilizing Berlekamp's architecture is shown in Fig. 3. The linear binary matrix has as its inputs the contents of the Z register. At a given time interval, the representation of a field element z in the dual basis is stored in register Z. The outputs of the matrix for a (255, 223) RS encoder (where J equals 8 and E equals 16) are

$$T_0 = Tr(zG_0)$$

$$T_1 = Tr(zG_1)$$

$$\vdots$$

$$T_{15} = Tr(zG_{15})$$

$$T_{16} = Tr(zG_{16})$$

For a given ℓ, $Tr(zG_\ell)$ is a parity check over a particular subset of the bits representing z in accordance with (17). These outputs represent

$$z_0^{(0)}, z_0^{(1)}, \cdots, z_0^{(15)}, z_0^{(16)}$$

the first components (bits) in the representation in the dual basis of the products

$$zG_0, zG_1, \cdots, zG_{15}, zG_{16}$$

respectively.

The output $Tr(\beta^i z)$ which is fed back to the Z register is used in deriving βz. A field element z may be represented as α^n or in the dual basis in vector form as

$$z = Tr(z), Tr(\beta z), \cdots, Tr(\beta^7 z)$$

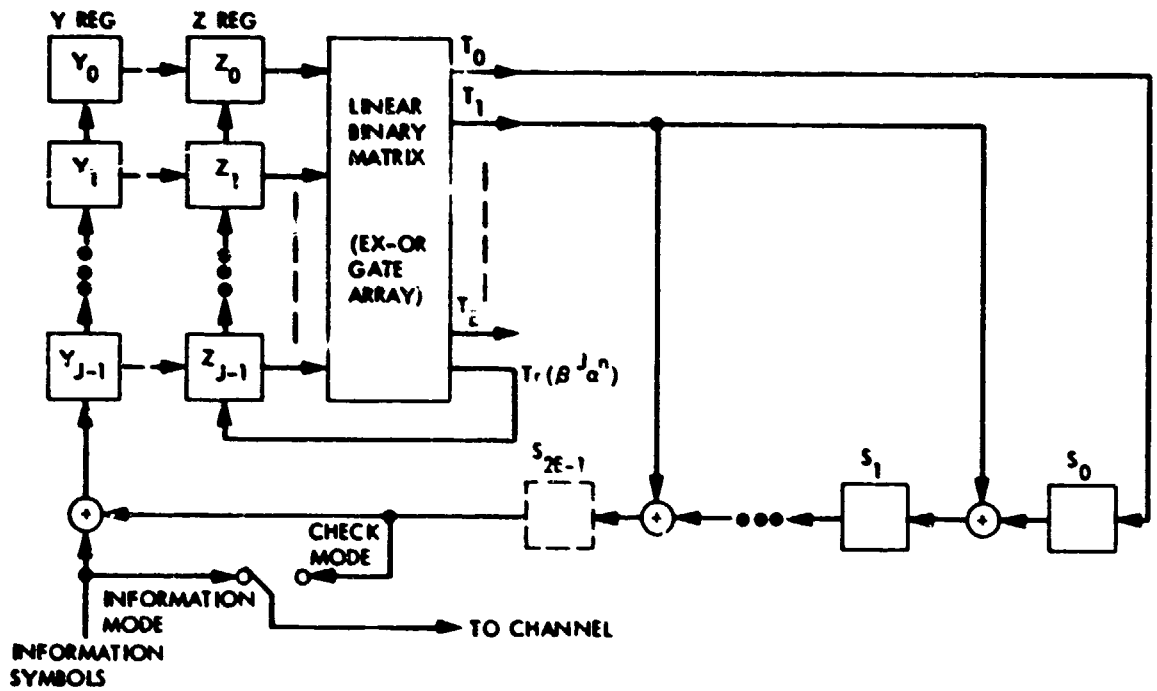

Figure 3. A (N,K) RS Encoder Utilizing Berlekamp's Architecture

where

$$Tr(\beta^i z) = Tr(\beta^i \alpha^n) = z_i$$

Thus

$$\beta z = Tr(\beta z), Tr(\beta^2 z), \cdots, Tr(\beta^8 z)$$

Computing βz from z corresponds to

$$z_i \leftarrow z_{i+1} \quad 0 \leq i < 7$$

$$z_7 \leftarrow Tr(\beta^8 z) = z_8$$

where the bits stored in the Z register are shifted and the output $Tr(\beta^8 z)$ of th binary matrix is entered. Clocking the Z register so configured yields βz, the set of inputs to the binary matrix during the subsequent time interval. The outputs

$$T_0 = Tr(\beta z G_0)$$

$$T_1 = Tr(\beta z G_1)$$

$$\vdots$$

$$T_{15} = Tr(\beta z G_{15})$$

$$T_{16} = Tr(\beta z G_{16})$$

represent $\{z_1^{(\ell)}\}$, the second components of $\{zG_\ell\}$, respectively. Similarily, t remaining components are computed recursively. The final components $\{z_1^{(\ell)}\}$ ar computed during the $8^{\underline{th}}$ time interval when $\beta^7 z$ resides in the Z register and th outputs are $\{T_\ell = Tr(\beta^7 z G_\ell)\}$.

Since $G_{32-\ell}$ equals G_ℓ

$$zG_{32-\ell} = zG_\ell \quad 1 \leq \ell \leq 16$$

The components of the products of

$$zG_{17}, zG_{18}, \cdots, zG_{31}$$

will have also been computed. The bit-serial multiplication of

$$G_{31}, G_{30}, \cdots, G_0$$

by z over $GF(2^8)$ is thus complete. Furthermore, the resultant vector

$$zG_{31}, zG_{30}, \cdots, zG_0$$

has been bit-serially added to the previous contents of the FSR (in Fig. 3), symbol-shifted one place to the left. Upon computing a set of corresponding components $\{z_i^{(\ell)}\}$, $z_i^{(0)}$ is entered into the register section S_0 as $z_i^{(1)}$, and $z_i^{(2)}, \cdots, z_i^{(31)}$ are each simultaneously Exclusive-ORed with the bit emanating from the register section $S_1, S_2, \cdots, S_{31}$, respectively. The field element z is a symbol (represented in the dual basis) being fed back during the encoding process.

Each register section except S_{31} is 40 bits in length and stores 5 8-bit symbols. This provides an interleaving depth of 5. Register Y serves as a staging register and is essentially an extension of S_{31}. After the products $\{zG_\ell\}$ have been determined, register Z is reloaded with the contents of register Y. At this time register Y contains the next symbol z to be fed back. Register sections $S_1, S_2, \cdots, S_{30}$ reside in RAM's. The Y and Z registers are composed of delay flip-flops and register section S_{31} is a serial shift register. Until all information symbols have been entered (and simultaneously delivered to the channel), the Y input is the bit-by-bit Exclusive-OR of the bits composing the information symbol being entered and the bits composing the symbol exiting register section S_{31}. After the last information symbol has been entered, a control

signal (not shown in Fig. 3) level is changed to disable the information input and switch from the information mode to the check mode. The 5 sets of 32 chec symbols are then bit-serially delivered to the channel as the Y and Z register and the S_i register sections are cleared.

The derivation of the functions $\{T_\ell\}$ is given in Example 6 for a (63, 53) RS code.

Example 6

Refer to Example 5 and Table 1, wherein every field element in $GF(2^6)$ is represented as

$$\alpha^n = u_5\alpha^5 + u_4\alpha^4 + \cdots + u_0$$

where $\alpha^6 = \alpha^5 + \alpha^2 + \alpha + 1$. For the basis $\{\beta^i\}$ in $GF(2^6)$

$$\{1, \beta, \beta^2, \beta^3, \beta^4, \beta^5\} = \{1, \alpha^3, \alpha^6, \alpha^9, \alpha^{12}, \alpha^{15}\}$$

the dual basis $\{\ell_j\}$ was shown to be

$$\{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4, \ell_5\} = \{\alpha^{48}, \alpha^{45}, \alpha^{26}, \alpha^{23}, \alpha^{43}, \alpha^{51}\}$$

An RS symbol is representable as α^n and in the dual basis as

$$z = z_0\ell_0 + z_1\ell_1 + \cdots + z_5\ell_5$$

where

$$z_i = \mathrm{Tr}(\beta^i\alpha^n) = \mathrm{Tr}(\alpha^{n+3i}).$$

It remains to select a self-reciprocal generator polynomial over GF(2) f the (63, 53) RS code where

$$J = 6, \quad N = 63, \quad E = 5, \quad I = 1$$

$$g(x) = \sum_{j=b}^{b+9} (x-\gamma^j) = \sum_{i=0}^{10} G_i x^i$$

From (11)

$$2b + 2E - 1 = N$$

$$2b + 9 = 63$$

$$b = 27 \text{ and } b + 9 = 36$$

The element γ where

$$\gamma^{27}, \gamma^{28}, \cdots, \gamma^{36}$$

are distinct roots of g(x) may be selected among $\phi(63)/2$ or 18 reciprocal pairs of primitive elements. The generator α is primitive and

$$\gamma = \alpha^k$$

is primitive if and only if $(k, 63) = 1$.

For $\gamma = \alpha^5$,

the coefficients of g(x) are

$$G_0 = G_{10} = 1 \qquad G_3 = G_7 = \alpha^{15}$$

$$G_1 = G_9 = \alpha^{34} \qquad G_4 = G_6 = 1$$

$$G_2 = G_8 = \alpha^4 \qquad G_5 = \alpha^{54}$$

The form of the $\{T_\ell\}$ functions (as shown in (17) for a (255, 223) RS code) is

$$T_\ell(z) = \mathrm{Tr}(zG_\ell) = \sum_{j=0}^{5} z_j \mathrm{Tr}(\ell_j G_\ell)$$

The values of the traces $\mathrm{Tr}(\ell_j G_\ell)$ are tabulated in Table 2. Values of $\mathrm{Tr}(\alpha^n)$ are given in Table 1.

Components z_j's of every z in the dual basis for which

$$\mathrm{Tr}(\ell_j G_\ell) = 1$$

contribute to the output T_ℓ. From Table 2, the T_ℓ functions are

$$T_0 = z_0 = T_4$$

$$T_1 = z_1 + z_5$$

$$T_2 = z_4 + z_5$$

$$T_3 = z_5$$

$$T_5 = z_2 + z_4 + z_5$$

The output

$$\mathrm{Tr}(\beta^6 \alpha^n) = z_6$$

required in deriving βz from z is determined as follows:

Table 2. $Tr(\ell_j G_\ell)$ Values for a (63, 53) RS Code

j	0	1	2	3	4	5
$\ell_j G_0$	α^{48}	α^{45}	α^{26}	α^{23}	α^{43}	α^{51}
$Tr(\ell_j G_0)$	1	0	0	0	0	0
$(\ell_j G_1)$	α^{19}	α^{16}	α^{60}	α^{57}	α^{14}	α^{22}
$Tr(\ell_j G_1)$	0	1	0	0	0	1
$\ell_j G_2$	α^{52}	α^{49}	α^{30}	α^{27}	α^{45}	α^{55}
$Tr(\ell_j G_2)$	0	0	0	0	1	1
$\ell_j G_3$	α^{0}	α^{60}	α^{41}	α^{38}	α^{58}	α^{3}
$Tr(\ell_j G_3)$	0	0	0	0	0	1
$\ell_j G_5$	α^{39}	α^{36}	α^{17}	14	α^{34}	α^{42}
$Tr(\ell_j G_5)$	0	0	1	0	1	1

$$\{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4, \ell_5\} = \{\alpha^{48}, \alpha^{45}, \alpha^{26}, \alpha^{23}, \alpha^{43}, \alpha^{51}\}$$

$$\{G_0, G_1, G_2, G_3, G_4, G_5\} = \{\alpha^{0}, \alpha^{34}, \alpha^{4}, \alpha^{15}, \alpha^{0}, \alpha^{54}\}$$

Note that $Tr(\ell_j G_0) = Tr(\ell_j G_4)$.

z_0	z_1	z_2	z_3	z_4	z_5	α^n	$\beta^6\alpha^n = \alpha^{n+18}$	$Tr(\beta^6\alpha^n)$
1	0	0	0	0	0	α^{48}	α^3	1
0	1	0	0	0	0	α^{45}	α^0	0
0	0	1	0	0	0	α^{26}	α^{44}	1
0	0	0	1	0	0	α^{23}	α^{41}	0
0	0	0	0	1	0	α^{43}	α^{61}	1
0	0	0	0	0	1	α^{51}	α^6	1

$$Tr(\beta^6\alpha^n) = z_6 = z_0 + z_2 + z_4 + z_5$$

The linear binary matrix with inputs $z_0, z_1, \cdots, z_5$ and outputs $T_0, T_1, \cdots, T_5$ and z_6 (i.e., $Tr(\beta^6\alpha^n)$) for a (63, 53) RS code is shown in Fig. 4. □

In a conventional (N,K) RS encoder, an information or check symbol is represented as

$$\alpha^n = u_0 + u_1\,\alpha + \cdots + u_{J-1}\,\alpha^{J-1}$$

and denoted by

$$u_0\,u_1 \cdots u_{J-1}$$

In an (N,K) RS encoder employing Berlekamp's architecture, the symbols (information and check) are represented in the dual basis. The transformation from one representation to the other is linear. The symbol α^i in the dual basis is represented as

$$\left[Tr(\alpha^i)\right]\,\ell_0 + \left[Tr(\beta\alpha^i)\right]\,\ell_1 + \cdots + \left[Tr(\beta^{J-1}\alpha^i)\right]\,\ell_{j-1}$$

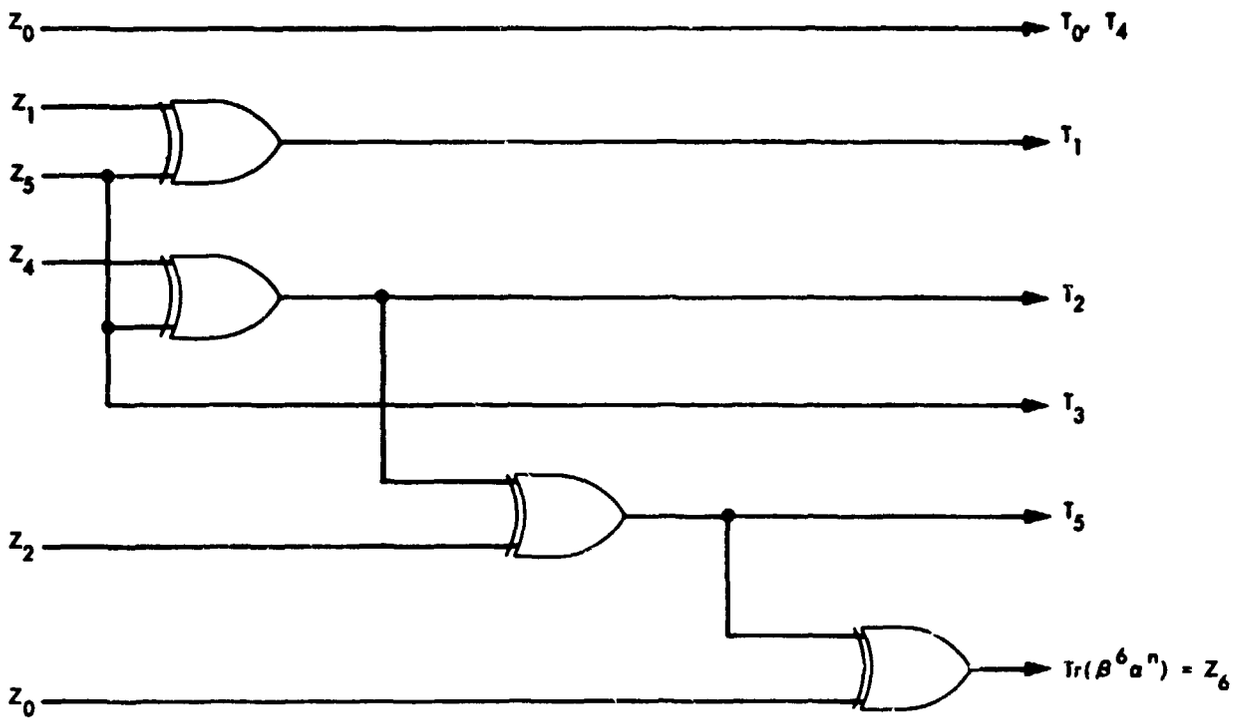

Figure 4. Implementation of the Linear Binary Matrix for a (63, 53) RS Code

and denoted by

$$z_0\, z_1 \cdots z_{J-1}$$

where $z_k = \mathrm{Tr}(\beta^k \alpha^i)$. Thus

$$\mathrm{Tr}(\alpha^i),\ \mathrm{Tr}(\beta\alpha^i),\ \cdots,\ \mathrm{Tr}(\beta^{J-1}\alpha^i) \leftrightarrow \alpha^i$$

$$\mathrm{Tr}(\alpha^j),\ \mathrm{Tr}(\beta\alpha^j),\ \cdots,\ \mathrm{Tr}(\beta^{J-1}\alpha^j) \leftrightarrow \alpha^j$$

and

$$\mathrm{Tr}(\alpha^i) + \mathrm{Tr}(\alpha^j),\ \mathrm{Tr}(\beta\alpha^i) + \mathrm{Tr}(\beta\alpha^j),\ \cdots,\ \mathrm{Tr}(\beta^{J-1}\alpha^i) + \mathrm{Tr}(\beta^{J-1}\alpha^j)$$

$$= \mathrm{Tr}\left[(\alpha^i+\alpha^j)\right],\ \mathrm{Tr}\left[\beta(\alpha^i+\alpha^j)\right],\ \cdots,\ \mathrm{Tr}\left[\beta^{J-1}(\alpha^i+\alpha^j)\right] \leftrightarrow \alpha^i + \alpha^j$$

The automorphism in $GF(2^J)$ of the two representations under the same rules of "addition" is illustrated in Example 7.

Example 7

Refer to the two representations of elements in $GF(2^6)$ in Table 1.

	u_5	u_4	u_3	u_2	u_1	u_0		z_0	z_1	z_2	z_3	z_4	z_5
α^{24}	1	1	0	0	1	0	$\leftrightarrow$	1	0	0	1	0	0
$+\ \alpha^{58}$	1	0	1	1	1	1	$\leftrightarrow$	0	1	1	1	0	1
α^{44}	0	1	1	1	0	1	$\leftrightarrow$	1	1	1	0	0	1

□

Example 8

Given the information symbol sequence

	ℓ_0	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5
C_{62}	0	0	0	0	0	0
.						
.						
.						
C_{11}	0	0	0	0	0	0
C_{10}	0	0	0	0	0	1

to be encoded by a (63, 53) RS encoder incorporating Berlekamp's architecture. Leading zero information symbols have no effect on the 10 6-bit register sections (which are initially cleared). The single nonzero information symbol is entered into the Z register via the Y register, and the T_ℓ functions (derived in Example 6 and implemented in Fig. 4) are applied to determine the z_0's of $\{zG_\ell\}$. Replacing z with βz (by clocking the Z register) and applying the T_ℓ functions yield the z_1's of $\{zG_2\}$ and so on as shown in Table 3.

The symbols of the codeword in Table 3 expressed as powers of α are compared with corresponding coefficients of g(x) as follows.

	C_{62}	$\cdots$	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
Codeword	α^*	$\cdots$	α^*	α^{51}	α^{22}	α^{55}	α^{3}	α^{51}	α^{42}	α^{51}	α^{3}	α^{55}	α^{22}	α^{51}
Coefficients of g(x)	α^*	$\cdots$	α^*	α^{0}	α^{34}	α^{4}	α^{15}	α^{0}	α^{54}	α^{0}	α^{15}	α^{4}	α^{34}	α^{0}

Note that the codeword polynomial is a scalar multiple (of α^{51}) of the g(x) of the (63, 53) RS code. This property of RS codes provides a simple check on the derived T_ℓ functions and z_6 (i.e., $\mathrm{Tr}(\beta^6\alpha^n)$). □

V. MATHEMATICAL CHARACTERIZATION OF THE (255, 223) RS ENCODER DESIGNED BY BERLEKAMP

As previously discussed, the independent parameter values of the (255, 223) RS code are

Table 3. Check Symbol Computation in the Dual Basis of a (63, 53) RS Code

	z	βz	$\beta^2 z$	$\beta^3 z$	$\beta^4 z$	$\beta^5 z$	ℓ_0	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	
							0	0	0	0	0	0	C_{62}
									⋮				
z_0	0	0	0	0	0	1	0	0	0	0	0	0	C_{11}
z_1	0	0	0	0	1	1	0	0	0	0	0	1	C_{10}
z_2	0	0	0	1	1	0	1	1	0	1	1	1	C_9
z_3	0	0	1	1	0	1	1	0	1	1	1	0	C_8
z_4	0	1	1	0	1	0	1	1	0	1	0	0	C_7
z_5	1	1	0	1	0	0	0	0	0	0	0	1	C_6
							1	0	1	0	0	0	C_5
							0	0	0	0	0	1	C_4
							1	1	0	1	0	0	C_3
							1	0	1	1	1	0	C_2
							1	1	0	1	1	1	C_1
							0	0	0	0	0	1	C_0
							↑					↑	
							$T_\ell(z)$					$T_\ell(\beta^5 z)$	

$T_0 = z_0 = T_4$ $\quad T_3 = z_5$

$T_1 = z_1 + z_5$ $\quad T_5 = z_2 + z_4 + z_5$

$T_2 = z_4 + z_5$ $\quad Tr(\beta^6 \alpha^n) = z_6 = z_0 + z_2 + z_4 + z_5$

J = 8 bits per symbol

E = 16 symbol error correction capability

I = 5 (symbol) interleaving depth

The generator α of the nonzero field elements in $GF(2^8)$ is a root of the primitive polynomial over GF(2)

$$x^8 + x^7 + x^2 + x + 1 \tag{18}$$

The field element

$$\beta = \alpha^{117} \tag{19}$$

was selected to form the basis in $GF(2^8)$

$$\{1, \beta, \beta^2, \cdots, \beta^7\} \quad \{\beta^i\}$$

The resulting dual basis is

$$\{\ell_j\} = \{\ell_0, \ell_1, \cdots, \ell_7\}$$

$$= \{\alpha^{125}, \alpha^{88}, \alpha^{226}, \alpha^{163}, \alpha^{46}, \alpha^{184}, \alpha^{67}, \alpha^{242}\} \tag{20}$$

The field element

$$\gamma = \alpha^{11} \tag{21}$$

was selected in specifying the self-reciprocal generator polynomial

$$g(x) = \prod_{j=112}^{143} (x-\gamma^j) = \sum_{i=0}^{32} G_i x^i$$

The coefficients of g(x) in expanded form are

$$
\begin{aligned}
G_0 = G_{32} &= \alpha^0 & \qquad G_8 = G_{24} &= \alpha^{97} \\
G_1 = G_{31} &= \alpha^{249} & \qquad G_9 = G_{23} &= \alpha^{30} \\
G_2 = G_{30} &= \alpha^{59} & \qquad G_{10} = G_{22} &= \alpha^{3} \\
G_3 = G_{29} &= \alpha^{66} & \qquad G_{11} = G_{21} &= \alpha^{213} \\
G_4 = G_{28} &= \alpha^{4} & \qquad G_{12} = G_{20} &= \alpha^{50} \\
G_5 = G_{27} &= \alpha^{43} & \qquad G_{13} = G_{19} &= \alpha^{66} \\
G_6 = G_{26} &= \alpha^{126} & \qquad G_{14} = G_{18} &= \alpha^{170} \\
G_7 = G_{25} &= \alpha^{251} & \qquad G_{15} = G_{17} &= \alpha^{5} \\
& & \qquad G_{16} &= \alpha^{24}
\end{aligned}
\tag{22}
$$

Note that $G_3 = G_{29} = G_{13} = G_{19}$.

The resulting T_ℓ functions are:

$$
\begin{aligned}
T_0 &= z_0 \\
T_1 &= z_1 + z_2 + z_4 + z_6 \\
T_2 &= z_2 + z_3 \\
T_3 &= z_0 + z_2 + z_3 + z_4 + z_5 = T_{13} \\
T_4 &= z_0 + z_2 + z_7 \\
T_5 &= z_0 + z_1 + z_2 + z_6 + z_7 \\
T_6 &= z_0 + z_1 + z_5 + z_6 \\
T_7 &= z_1 + z_2 + z_4 \\
T_8 &= z_0 + z_1 + z_3 + z_6 + z_7 \\
T_9 &= z_0 + z_2 + z_3 + z_4 + z_5 \\
T_{10} &= z_0 + z_1 + z_4 + z_7 \\
T_{11} &= z_4
\end{aligned}
$$

$$T_{12} = z_0 + z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7$$
$$T_{14} = z_0 + z_1 + z_2 \quad + z_4 + z_5 + z_6$$
$$T_{15} = z_0 + z_1 \quad + z_3 \quad + z_5 \quad + z_7$$
$$T_{16} = \quad z_1 + z_2 \quad + z_6$$

In addition to the T_ℓ functions

$$Tr(\beta^8 \alpha^n) = z_8 = z_0 + z_1 + z_3 + z_7$$

is an output of the linear binary matrix as discussed in Section IV-B.

The sole criterion in the selection of β in (19) and γ in (21) was the realization of a linear binary matrix of minimal complexity. The dual basis $\{\ell_j\}$ directly results from the selection of β. The coefficients $\{G_\ell\}$ of g(x) are fixed by the choice of γ. As discussed and shown in (17), those components z_j's in the dual basis for which

$$Tr(\ell_j G_\ell) = 1$$

contribute to the output T_ℓ. A measure of complexity for a given β and γ is the number of 1's in the set

$$\{Tr(\ell_j G_\ell)\} \text{ for } 0 \leq j < 8$$

and distinct G_ℓ's among

$$\{G_0, G_1, \cdots, G_{16}\}$$

Using this measure, Berlekamp combined a computer search with some hand computation in finding a $\beta\gamma$ combination yielding a set of T_ℓ functions of minimal complexity. The entire binary matrix was realized with 24 2-input Exclusive-OR gates organized for maximum gate sharing within three levels of gating.

The two representations of field elements in $GF(2^8)$ appear in Table 4.

Table 4. Two Representations of Field Elements in GF(2^8)

n of α^n	i of α^i	Tr(α^n)	j of ℓ_j	n of α^n	i of α^i	Tr(α^n)	j of ℓ_j
	76543210		01234567		76543210		01234567
*	00000000	0	00000000	46	11110000	0	00001000(ℓ_4)
0	00000001	0	01111011	47	01100111	0	01001110
1	00000010	1	10101111	48	11001110	1	10101110
2	00000100	1	10011001	49	00011011	1	10101000
3	00001000	1	11111010	50	00110110	0	01011100
4	00010000	1	10000110	51	01101100	0	01100000
5	00100000	1	11101100	52	11011000	0	00011110
6	01000000	1	11101111	53	00110111	0	00100111
7	10000000	1	10001101	54	01101110	1	11001111
8	10000111	1	11000000	55	11011100	1	10000111
9	10001001	0	00001100	56	00111111	1	11011101
10	10010101	1	11101001	57	01111110	0	01001001
11	10101101	0	01111001	58	11111100	0	01101011
12	11011101	1	11111100	59	01111111	0	00110010
13	00111101	0	01110010	60	11111110	1	11000100
14	01111010	1	11010000	61	01111011	1	10101011
15	11110100	1	10010001	62	11110110	0	00111110
16	01101111	1	10110100	63	01101011	0	00101101
17	11011110	0	00101000	64	11010110	1	11010010
18	00111011	0	01000100	65	00101011	1	11000010
19	01110110	1	10110011	66	01010110	0	01011111
20	11101100	1	11101101	67	10101100	0	00000010(ℓ_6)
21	01011111	1	11011110	68	11011111	0	01010011
22	10111110	0	00101011	69	00111001	1	11101011
23	11111011	0	00100110	70	01110010	0	00101010
24	01110001	1	11111110	71	11100100	0	00010111
25	11100010	0	00100001	72	01001111	0	01011000
26	01000011	0	00111011	73	10011110	1	11000111
27	10000110	1	10111011	74	10111011	1	11001001
28	10001011	1	10100011	75	11110001	0	01110011
29	10010001	0	01110000	76	01100101	1	11100001
30	10100101	1	10000011	77	11001010	0	00110111
31	11001101	0	01111010	78	00010011	0	01010010
32	00011101	1	10011110	79	00100110	1	11011010
33	00111010	0	00111111	80	01001100	1	10001100
34	01110100	0	00011100	81	10011000	1	11110001
35	11101000	0	01110100	82	10110111	1	10101010
36	01010111	0	00100100	83	11101001	0	00001111
37	10101110	1	10101101	84	01010101	1	10001011
38	11011011	1	11001010	85	10101010	0	00110100
39	00110001	0	00010001	86	11010011	0	00110000
40	01100010	1	10101100	87	00100001	1	10010111
41	11000100	1	11111011	88	01000010	0	01000000(ℓ_1)
42	00001111	1	10110111	89	10000100	0	00010100
43	00011110	0	01001010	90	10001111	0	00111010
44	00111100	0	00001001	91	10011001	1	10001010
45	01111000	0	01111111	92	10110101	0	00000101

Table 4. Two Representations of Field Elements in GF(2^8) (contd)

n of α^n	i of α^i	Tr(α^n)	j of ℓ_j	n of α^n	i of α^i	Tr(α^n)	j of ℓ_j
	76543210		01234567		76543210		01234567
93	11101101	1	10010110	140	11001011	0	01001100
94	01011101	0	01110001	141	00010001	1	11111101
95	10111010	1	10110010	142	00100010	0	01000011
96	11110011	1	11011100	143	01000100	0	01110110
97	01100001	0	01111000	144	10001000	0	01110111
98	11000010	1	11001101	145	10010111	0	01000110
99	00000011	1	11010100	146	10101001	1	11100000
100	00000110	0	00110110	147	11010101	0	00000110
101	00001100	0	01100011	148	00101101	1	11110100
102	00011000	0	01111100	149	01011010	0	00111100
103	00110000	0	01101010	150	10110100	0	01111110
104	01100000	0	00000011	151	11101111	0	00111001
105	11000000	0	01100010	152	01011001	1	11101000
106	00000111	0	01001101	153	10110010	0	01001000
107	00001110	1	11001100	154	11100011	0	01011010
108	00011100	1	11100101	155	01000001	1	10010100
109	00111000	1	10010000	156	10000010	0	00100010
110	01110000	1	10000101	157	10000011	0	01011001
111	11100000	1	10001110	158	10000001	1	11110110
112	01000111	1	10100010	159	10000101	0	01101111
113	10001110	0	01000001	160	10001101	1	10010101
114	10011011	0	00100101	161	10011101	0	00010011
115	10110001	1	10011100	162	10111101	1	11111111
116	11100101	0	01101100	163	11111101	0	00010000(ℓ_3)
117	01001101	1	11110111	164	01111101	1	10011101
118	10011010	0	01011110	165	11111010	0	01011101
119	10110011	0	00110011	166	01110011	0	01010001
120	11100001	1	11110101	167	11100110	1	10111000
121	01000101	0	00001101	168	01001011	1	11000001
122	10001010	1	11011000	169	10010110	0	00111101
123	10010011	1	11011111	170	10101011	0	01001111
124	10100001	0	00011010	171	11010001	1	10011111
125	11000101	1	10000000(ℓ_0)	172	00100101	0	00001110
126	00001101	0	00011000	173	01001010	1	10111010
127	00011010	1	11010011	174	10010100	1	10010010
128	00110100	1	11110011	175	10101111	1	11010110
129	01101000	1	11111001	176	11011001	0	01100101
130	11010000	1	11100100	177	00110101	1	10001000
131	00100111	1	10100001	178	01101010	0	01010110
132	01001110	0	00100011	179	11010100	0	01111101
133	10011100	0	01101000	180	00101111	0	01011011
134	10111111	0	01010000	181	01011110	1	10100101
135	11111001	1	10001001	182	10111100	1	10000100
136	01110101	0	01100111	183	11111111	1	10111111
137	11101010	1	11011011	184	01111001	0	00000100(ℓ_5)
138	01010011	1	10111101	185	11110010	1	10100111
139	10100110	0	01010111	186	01100011	1	11010111

Table 4. Two Representations of Field Elements in GF(2^8) (contd)

n of α^n	i of α^i	Tr(α^n)	j of ℓ_j	n of α^n	i of α^i	Tr(α^n)	j of ℓ_j
	76543210		01234567		76543210		01234567
187	11000110	0	01010100	234	10001100	1	11101110
188	00001011	0	00101110	235	10011111	1	10111100
189	00010110	1	10110000	236	10111001	0	01100110
190	00101100	1	10001111	237	11110101	1	11101010
191	01011000	1	10010011	238	01101101	0	00011011
192	10110000	1	11100111	239	11011010	1	10110001
193	11100111	1	11000011	240	00110011	1	10111110
194	01001001	0	01101110	241	01100110	0	00110101
195	10010010	1	10100100	242	11001100	0	00000001(ℓ_7)
196	10100011	1	10110101	243	00011111	0	00110001
197	11000001	0	00011001	244	00111110	1	10100110
198	00000101	1	11100010	245	01111100	1	11100110
199	00001010	0	01010101	246	11111000	1	11110010
200	00010100	0	00011111	247	01110111	1	11001000
201	00101000	0	00010110	248	11101110	0	01000010
202	01010000	0	01101001	249	01011011	0	01000111
203	10100000	0	01100001	250	10110110	1	11010001
204	11000111	0	00101111	251	11101011	1	10100000
205	00001001	1	10000001	252	01010001	1	00010010
206	00010010	0	00101001	253	10100010	1	11001110
207	00100100	0	01110101	254	11000011	1	10110110
208	01001000	0	00010101				
209	10010000	0	00001011				
210	10100111	0	00101100				
211	11001001	1	11100011				
212	00010101	0	01100100				
213	00101010	1	10111001				
214	01010100	1	11110000				
215	10101000	1	10011011				
216	11010111	1	10101001				
217	00101001	0	01101101				
218	01010010	1	11000110				
219	10100100	1	11111000				
220	11001111	1	11010101				
221	00011001	0	00000111				
222	00110010	1	11000101				
223	01100100	1	10011010				
224	11001000	1	10011000				
225	00010111	1	11001011				
226	00101110	0	00100000(ℓ_2)				
227	01011100	0	00001010				
228	10111000	0	00011101				
229	11110111	0	01000101				
230	01101001	1	10000010				
231	11010010	0	01001011				
232	00100011	0	00111000				
233	01000110	1	11011001				

VI. HARDWARE COST OF RS ENCODERS — CONVENTIONAL VS BERLEKAMP'S ARCHITECTURE

There are two existing designs utilizing Galileo flight-qualified parts with enough similarity in their functional specifications to make a meaningful comparison.

(1) One is a conventional (255, 223) RS encoder with an interleaving depth I of 2. It accepts a bit-serial input of up to approximately 800 kbits per second. An input sequence comprised of 2 sets of 223 8-bit symbols (i.e., 3568 bits) need not be continuous. This encoder will serve as an outer encoder for compressed imaging data aboard the Galileo spacecraft.

(2) The other is a (255, 223) RS encoder utilizing Berlekamp's architecture with an interleaving depth I of 5. The (255I, 223I) code can be shortened to a ((255-Q)I, (223-Q)I) shortened code where

$$223 - Q \geq 1$$

The leading QI symbols of the shortened code are viewed as 0's (00...0) and discarded. It accepts a bit-serial input up to approximately 400 kbits per second. An input sequence comprised of 5 sets of 223-Q 8-bit symbols need not be continuous. This encoder was designed and implemented in breadboard form by E.R. Berlekamp of Cyclotomics Inc. under a JPL contract. It has been adopted as the outer encoder for all science and engineering data emanating from the NASA ISPM spacecraft. The JPL specifications were in accordance with Galileo requirements, which exceed those of ISPM (specifically serial input and output bit rates). The (255, 223) RS code with an interleaving depth of 5 was a contender to the (24, 12) extended (binary) Golay code (bit) interleaved to a depth of 36. The extended Golay code was the early choice for the outer code of nonimaging science data for the Galileo spacecraft. Subsequent to a third and final review, the extended Golay code will serve as the outer code. However, packetized telemetry with RS/convolutional concatenated coding has been adopted as a NASA-JPL standard for future spacecraft missions.

The logic building blocks used in (1) and (2) were integrated circuits (ICs) in the Complementary-symmetry Metal Oxide Semiconductor (CMOS) family. Low power and amenability to radiation hardening are characteristics of CMOS technology that are essential in space applications. ROMs and RAMs each occupy 3 16-pin IC locations on flight circuit boards. The number of ICs and IC locations for each design are as follows:

(1) Conventional RS encoder

Total number of ICs excluding ROMs and RAMs	26
Total number of ROMs and RAMs	24
Total number of 16 pin locations	98

(2) RS encoder with Berlekamp architecture

Total number of ICs excluding RAMs	31
Total number of RAMs	8
Total number of 16 pin locations	5.

VII. TESTING RS ENCODERS

A. INTRODUCTION

As described in Section IV, a conventional RS encoder contains an FSR. If symbol interleaving is required, each register section of the FSR is lengthened by a factor of I (see Eq. 10). An RS encoder utilizing Berlekamp's architecture similarly incorporates an FSR. It differs principally from a conventional encoder in that symbol multiplication is bit-serial and is realized in hardware.

The size of the codeword dictionary of a (255, 223) RS code is $(256)^{223}$ or 2^{1784} (which approximately equals 10^{537}). However, the number of information symbol sequences required to test the functional integrity of an encoder is surprisingly small. This is due to the linearity, cyclic structure and other properties (subsequently discussed) of RS codes.

Three classes of RS symbol sequences provide a user with a simple, systematic and effective means of testing conventional as well as Berlekamp types of RS encoders. Hereafter, the classes of symbol sequences are referred to as the generator polynomial coefficient sequence (GCS), the constant symbol sequence (CS), and the iterative symbol sequence (IS).

Conventional and Berlekamp type of (255, 223) RS encoders are assumed to have the same self-reciprocal generator polynomial whose coefficients appear in (22).

The representation of symbols associated with the conventional encoder are the polynomials in α appearing in Table 4. Corresponding to each polynomial in α is the representation in the dual basis of symbols associated with the Berlekamp-type encoder. Given

$$\alpha^n = u_7\alpha^7 + u_6\alpha^6 + \cdots + u_0$$

the corresponding element is

$$z = z_0\ell_0 + z_1\ell_1 + \cdots + z_7\ell_7$$

where

$$[z_0, z_1, \cdots, z_7] = [u_7, u_6, \cdots, u_0]\, T_{\alpha\ell}$$

and

$$T_{\alpha\ell} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{24}$$

Row 1, row 2, $\cdots$, and row 8 in T are representations in the dual basis of α^7 (10 $\cdots$, 0), α^6 (010 $\cdots$ 0), $\cdots$, and α^0 (00 $\cdots$ 01), respectively. The inverse of $T_{\alpha\ell}$ is

$$T_{\alpha\ell}^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \qquad (25)$$

Row 1, row 2, $\cdots$, and row 8 in $T_{\alpha\ell}^{-1}$ are polynomials in α corresponding to ℓ_0 (10 $\cdots$ 0), ℓ_1 (010 $\cdots$ 0), $\cdots$, and ℓ_7 (00 $\cdots$ 01), respectively. Thus

$$[z_0, z_1, \cdots, z_7]\, T_{\alpha\ell}^{-1} = [u_7, u_6, \cdots, u_0]$$

Given a conventional and a Berlekamp type of an (N,K) RS encoder with a common g(x). The transformational equivalence of codewords is illustrated in Fig. 5.

B. TEST SEQUENCES

1. The Generator (Polynomial) Coefficient Sequence (GCS)

The generator polynomial and every scalar multiple of the generator polynomial of an RS code are codeword polynomials of lowest degree. Consider the information symbol sequence

$$C_{254}\ C_{253}\ \cdots\ C_{33}\ C_{32}$$

$$\alpha^*\ \alpha^*\ \cdots\ \alpha^*\ \alpha^0$$

associated with a conventional (255, 223) RS encoder. Only the last information symbol C_{32} is nonzero. Thus

$$I(x) = \alpha^0\ (00 \cdots 01)$$

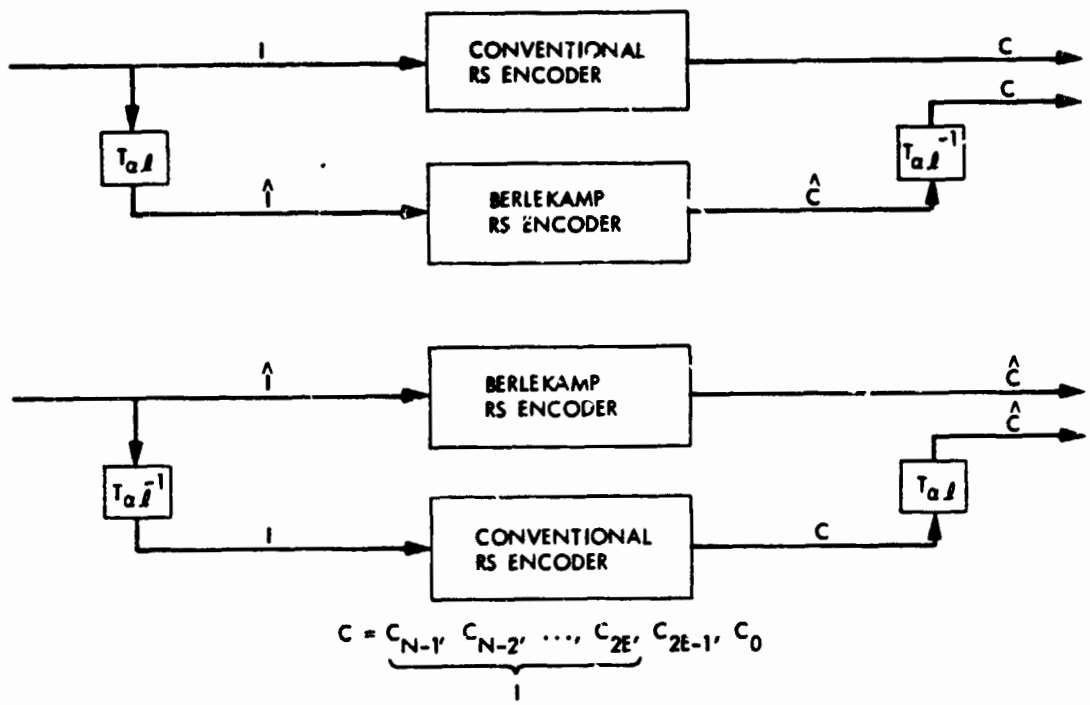

Figure 5. Transformational Equivalence of RS Codewords with a Common g(x)

and

$$x^{32}I(x) \equiv \sum_{i=0}^{31} G_i x^i \bmod g(x)$$

where $G_{31}, G_{30}, \cdots, G_0$ as given in (22) are the check symbols. The encoded word is

$$C_{32}\ C_{31}\ C_{30}\ C_{29} \cdots C_3\ C_2\ C_1\ C_0$$

$$\alpha^0\ \alpha^{249}\ \alpha^{59}\ \alpha^{66} \cdots \alpha^{66}\ \alpha^{59}\ \alpha^{249}\ \alpha^0$$

the GCS where the leading all-zeros symbols are not shown.

Every codeword polynomial

$$C(x) = C_{254}\, x^{254} + C_{253}\, x^{253} + \cdots + C_1 x + C_0$$

contains g(x) as a factor and is a member of a principal ideal of the ring F[x] / $(x^{255}-1)$ over GF(2^8) where g(x) is a generator of the ideal (see Ref. 2). Thus

$$xC(x) = C_{253}\, x^{254} + \cdots + C_1\, x^2 + C_0 x + C_{254}\, x^{255}$$

$$= C_{253}\, x^{254} + \cdots + C_1\, x^2 + C_0 x + C_{254}$$

since

$$x^{255} \equiv 1 \bmod x^{255}-1.$$

The coefficients of xC(x) are a cyclic permutation one place to the left of those of C(x). It follows that cyclic permutations of a GCS are codewords. Furthermore, linear combinations of GCSs and cyclic permutations of GCSs are also codewords.

Example 9

The results of a GCS-type test on a Berlekamp-type encoder with symbol interleaving to a depth of 5 appear in Table 5. Each of the 5 columns of 8-bit symbols is a codeword. The encoder is initially cleared and the 5 sets of leading information symbols (C_{254} through C_{33}) of all zeros are not shown. Information symbols enter and check symbols exit row by row as shown in the deinterleaved arrangement of Table 5. Note that C_{32} of codeword 5 is the only nonzero information symbol. The information symbol C_{32} with the resulting check symbols (C_{31} through C_0) is a representation of a scalar multiple of the GCS in the dual basis. From (25) and (22),

$$[C_{32}, C_{31}, \cdots, C_0]\, T_{\alpha\ell}^{-1} = \alpha^{-93}\, [G_{32}, G_{31}, \cdots, G_0]$$

Successive applications of the T_ℓ functions in (23) on z (11 $\cdots$ 1), βz, $\cdots$, $\beta^7 z$ (for β in (19)) yield the like components $\{z_0^{(\ell)}\}$, $\{z_1^{(\ell)}\}$, $\cdots$, $\{z_7^{(\ell)}\}$ of the symbols C_{32}, C_{31}, $\cdots$, C_0 of codeword 5 in Table 5. As in Table 3 for Example 8, the GSC provides a simple check on the derived T_ℓ functions and z_8 (i.e., $\mathrm{Tr}(\beta^8 \alpha^n)$).

□

Example 10

A test run on a Berlekamp encoder resulting in a cyclic permutation of the GCS in Example 9 is given in Table 6. The information symbols C_{33} and C_{32} of codeword 5 result in a GCS that is cyclically shifted upward one symbol. C_{254} (α^*) in Example 9 is C_0 in Example 10. The leading 222 information symbols in this example are all zeros.

Clearly, an all-zeros symbol (information and check) sequence is a codeword (e.g., codewords 1 through 4). It is the identity element of the linear (code) space and is representable as $\alpha^* g(x)$. □

Table 5. GCS Test for a Berlekamp Encoder

Codeword					
1	2	3	4	5	
				z_0 z_7	
00000000	00000000	00000000	00000000	11111111	C_{32}
00000000	00000000	00000000	00000000	00100010	
00000000	00000000	00000000	00000000	00000111	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	01010001	
00000000	00000000	00000000	00000000	10000001	
00000000	00000000	00000000	00000000	00111111	
00000000	00000000	00000000	00000000	11110110	
00000000	00000000	00000000	00000000	10000110	
00000000	00000000	00000000	00000000	11100111	
00000000	00000000	00000000	00000000	01011101	
00000000	00000000	00000000	00000000	11110101	
00000000	00000000	00000000	00000000	01100100	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	00110111	
00000000	00000000	00000000	00000000	10111000	
00000000	00000000	00000000	00000000	11010111	
00000000	00000000	00000000	00000000	10111000	
00000000	00000000	00000000	00000000	00110111	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	01100100	
00000000	00000000	00000000	00000000	11110101	
00000000	00000000	00000000	00000000	01011101	
00000000	00000000	00000000	00000000	11100111	
00000000	00000000	00000000	00000000	10000110	
00000000	00000000	00000000	00000000	11110110	
00000000	00000000	00000000	00000000	00111111	
00000000	00000000	00000000	00000000	10000001	
00000000	00000000	00000000	00000000	01010001	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	00000111	
00000000	00000000	00000000	00000000	00100010	
00000000	00000000	00000000	00000000	11111111	C_0

Table 6. A Cyclic Permutation of the GCS in Table 5

Codeword					
1	2	3	4	5	
				z_0 z_7	
00000000	00000000	00000000	00000000	11111111	c_{33}
00000000	00000000	00000000	00000000	00100010	c_{32}
00000000	00000000	00000000	00000000	00000111	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	01010001	
00000000	00000000	00000000	00000000	10000001	
00000000	00000000	00000000	00000000	00111111	
00000000	00000000	00000000	00000000	11110110	
00000000	00000000	00000000	00000000	10000110	
00000000	00000000	00000000	00000000	11100111	
00000000	00000000	00000000	00000000	01011101	
00000000	00000000	00000000	00000000	11110101	
00000000	00000000	00000000	00000000	01100100	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	00110111	
00000000	00000000	00000000	00000000	10111000	
00000000	00000000	00000000	00000000	11010111	
00000000	00000000	00000000	00000000	10111000	
00000000	00000000	00000000	00000000	00110111	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	01100100	
00000000	00000000	00000000	00000000	11110101	
00000000	00000000	00000000	00000000	01011101	
00000000	00000000	00000000	00000000	11100111	
00000000	00000000	00000000	00000000	10000110	
00000000	00000000	00000000	00000000	11110110	
00000000	00000000	00000000	00000000	00111111	
00000000	00000000	00000000	00000000	10000001	
00000000	00000000	00000000	00000000	01010001	
00000000	00000000	00000000	00000000	00011101	
00000000	00000000	00000000	00000000	00000111	
00000000	00000000	00000000	00000000	00100010	
00000000	00000000	00000000	00000000	11111111	
00000000	00000000	00000000	00000000	00000000	c_0

Example 11

Tables 7 and 8 present test runs on a Berlekamp encoder whereby C_{32} of 8 GCS type tests are the unit vectors (10 ··· 0), (010 ··· 0), ···, (00 ··· 01), respectively, in the dual basis. "Adding" corresponding symbols yields the (scalar multiple of the) GSC in Table 5. The code's generator matrix can be obtained from linear combinations of cyclic permutations of the GCS-type sequences. Each of the 223 distinct codewords in the code's generator matrix has one and only one nonzero information symbol. □

Consider the check symbol sequences for 20 information symbol sequences where each information symbol is randomly selected. Those computed by the encoder could be compared with those derived from the code's generator matrix. A symbol-by-symbol match for each of the 20 pairs of check symbol sequences would verify the functional integrity of an RS encoder with an extremely high degree of confidence. Additional tests serve to increase that degree of confidence.

2. The Constant (Symbol) Sequence (CS)

The polynomial $x^{255}-1$ factors are as follows.

$$x^{255}-1 = (x-1)(x^{254} + x^{253} + \cdots + x + 1)$$

Since the roots of g(x) are among the 255 roots of unity (i.e., the nonzero elements of $GF(2^8)$), g(x) divides $x^{255}-1$. Since

$$(g(x), x-1) = 1$$

g(x) must divide the factor of degree 254. Thus f(x) and any scalar multiplie Kf(x) where

$$f(x) = \sum_{i=0}^{254} x^i \qquad \text{and } K \in GF(2^8)$$

are codewords. The information symbol sequence

Table 7. CCS Type Tests with C_{32} One Set of Unit Vectors in the Dual Basis

Codeword					
1	2	3	4	5	
			z_0 z_7		
00000001	00000010	00000100	00001000	00000000	C_{32}
01100110	10101011	01010110	10101100	00000000	
00001000	00011000	00110001	01100010	00000000	
00100111	01101000	11010001	10100010	00000000	
11110010	00010111	00101110	01011100	00000000	
10000011	10000101	00001010	00010100	00000000	
01000001	11000011	10000111	00001110	00000000	
00011011	00101101	01011011	10110111	00000000	
10001011	10011101	00111011	01110110	00000000	
00101000	01111000	11110000	11100001	00000000	
11100110	00101010	01010100	10101000	00000000	
00011111	00100001	01000011	10000110	00000000	
10101101	11110111	11101110	11011100	00000000	
00100111	01101000	11010001	10100010	00000000	
01011001	11101010	11010100	10101001	00000000	
11001000	01011000	10110000	01100000	00000000	
01111001	10001010	00010101	00101010	00000000	
11001000	01011000	10110000	01100000	00000000	
01011001	11101010	11010100	10101001	00000000	
00100111	01101000	11010001	10100010	00000000	
10101101	11110111	11101110	11011100	00000000	
00011111	00100001	01000011	10000110	00000000	
11100110	00101010	01010100	10101000	00000000	
00101000	01111000	11110000	11100001	00000000	
10001011	10011101	00111011	01110110	00000000	
00011011	00101101	01011011	10110111	00000000	
01000001	11000011	10000111	00001110	00000000	
10000011	10000101	00001010	00010100	00000000	
11110010	00010111	00101110	01011100	00000000	
00100111	01101000	11010001	10100010	00000000	
00001000	00011000	00110001	01100010	00000000	
01100110	10101011	01010110	10101100	00000000	
00000001	00000010	00000100	00001000	00000000	C_0

Table 8. GCS Type Tests with C_{32} a Second Set of Unit Vectors in the Dual Basis

Codeword					
1	2	3	4	5	
			z_0 z_7		
00010000	00100000	01000000	10000000	00000000	C_{32}
01011001	11010101	10101010	00110011	00000000	
11000101	10000011	00000110	00000100	00000000	
01000101	10101101	01011010	10010011	00000000	
10111000	10000010	00000101	11111001	00000000	
00101001	11010000	10100001	11000001	00000000	
00011100	01111000	11110000	10100000	00000000	
01101111	11000101	10001011	00001101	00000000	
11101100	01010011	10100111	11000101	00000000	
11000011	10101111	01011110	10010100	00000000	
01010001	01000101	10001010	11110011	00000000	
00001101	00000100	00001000	00001111	00000000	
10111001	11011110	10111101	11010110	00000000	
01000101	10101101	01011010	10010011	00000000	
01010010	11111101	11111010	10101100	00000000	
11000001	01001011	10010110	11100100	00000000	
01010100	11010001	10100010	00111100	00000000	
11000001	01001011	10010110	11100100	00000000	
01010010	11111101	11111010	10101100	00000000	
01000101	10101101	01011010	10010011	00000000	
10111001	11011110	10111101	11010110	00000000	
00001101	00000100	00001000	00001111	00000000	
01010001	01000101	10001010	11110011	00000000	
11000011	10101111	01011110	10010100	00000000	
11101100	01010011	10100111	11000101	00000000	
01101111	11000101	10001011	00001101	00000000	
00011100	01111000	11110000	10100000	00000000	
00101001	11010000	10100001	11000001	00000000	
10111000	10000010	00000101	11111001	00000000	
01000101	10101101	01011010	10010011	00000000	
11000101	10000011	00000110	00000100	00000000	
01011001	11010101	10101010	00110011	00000000	
00010000	00100000	01000000	10000000	00000000	C_0

$$C_{254}\ C_{253}\ \cdots\ C_{32}$$

$$K\quad K\quad \cdots\ K$$

results in check symbols C_{31} through C_0 that are also equal to K.

In Table 9, codewords 2 through 5 are CS test sequences. Note that codeword 1 added to codeword 5 in Table 5 cyclically shifted downward one symbol is a CS of 11 $\cdots$ 1's in the dual basis.

3. Iterative (Symbol) Sequences (ISs)

Over every field x^d-1 divides x^n-1 if and only if d divides n. In any field which contains a primitive $n^{\underline{th}}$ root of unity

$$x^n-1 = \sum_{i=0}^{n-1} (x-\alpha^i)$$

as discussed in Ref. 1. Also if n = kd, then

$$\alpha^0,\ \alpha^k,\ \alpha^{2k},\ \cdots,\ \alpha^{(d-1)k}$$

are roots of

$$x^d-1 = 0$$

Consider the polynomials

$$Q_d(x) = \frac{x^{255}-1}{x^d-1} = \sum_{j=0}^{(255/d)-1} (x^d)^j$$

where d divides $255 = 3\cdot5\cdot17$. Since

$$x^{255}-1 = (x^d-1)\ Q_d(x)$$

Table 9. One Nonconstant and Four CS Type Tests Applied to a Berlekamp Encoder

	Codeword				
1	2	3	4	5	
z_0 z_7					
00000000	11111111	11111111	11111111	11111111	C_{254}
11111111	11111111	11111111	11111111	11111111	
11111111	11111111	11111111	11111111	11111111	
11111111	11111111	11111111	11111111	11111111	
...	...	...	...	...	
11111111	11111111	11111111	11111111	11111111	C_{32}
00000000	11111111	11111111	11111111	11111111	
11011101	11111111	11111111	11111111	11111111	
11111000	11111111	11111111	11111111	11111111	
11100010	11111111	11111111	11111111	11111111	
10101110	11111111	11111111	11111111	11111111	
01111110	11111111	11111111	11111111	11111111	
11000000	11111111	11111111	11111111	11111111	
00001001	11111111	11111111	11111111	11111111	
01111001	11111111	11111111	11111111	11111111	
00011000	11111111	11111111	11111111	11111111	
10100010	11111111	11111111	11111111	11111111	
00001010	11111111	11111111	11111111	11111111	
10011011	11111111	11111111	11111111	11111111	
11100010	11111111	11111111	11111111	11111111	
11001000	11111111	11111111	11111111	11111111	
01000111	11111111	11111111	11111111	11111111	
00101000	11111111	11111111	11111111	11111111	
01000111	11111111	11111111	11111111	11111111	
11001000	11111111	11111111	11111111	11111111	
11100010	11111111	11111111	11111111	11111111	
10011011	11111111	11111111	11111111	11111111	
00001010	11111111	11111111	11111111	11111111	
10100010	11111111	11111111	11111111	11111111	
00011000	11111111	11111111	11111111	11111111	
01111001	11111111	11111111	11111111	11111111	
00001001	11111111	11111111	11111111	11111111	
11000000	11111111	11111111	11111111	11111111	
01111110	11111111	11111111	11111111	11111111	
10101110	11111111	11111111	11111111	11111111	
11100010	11111111	11111111	11111111	11111111	
11111000	11111111	11111111	11111111	11111111	
11011101	11111111	11111111	11111111	11111111	

g(x) divides $Q_d(x)$ if g(x) and x^d-1 have no common roots (i.e., $(g(x), x^d-1) = 1$). The roots $\{R_d\}$ of

$$x^d-1 = 0$$

for various values of d are

$$\{R_3\} = \{\alpha^i : i = 0, 85, 170\}$$

$$\{R_5\} = \{\alpha^i : i = 0, 51, 102, 153, 204\}$$

$$\{R_{15}\} = \{\alpha^i : i = 0, 17, 34, 51, 68, 85, 102, 119, 136, 153, 170, 187, 204, 221, 238\}$$

$$\{R_{17}\} = \{\alpha^i : i = 0, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240\}$$

The roots of g(x) whose coefficients are given in (22) are

$$\{R_g\} = \{(\alpha^{11})^{j \bmod 255} : 112 \leq j \leq 143\}$$

or (in ascending powers of α)

$$= \{\alpha^i : i = 1, 10, 12, 21, 23, 32, 34, 43, 45, 56, 67, 78, 89, 100, 111, 122, 133, 144, 155, 166, 177, 188, 199, 210, 212, 221, 223, 232, 234, 243, 245, 254\}$$

Since $\{R_3\}$ and $\{R_g\}$ have no common elements g(x) divides

$$Q_3(x) = \frac{x^{255}-1}{x^3-1} = \sum_{j=0}^{84} x^{3j}$$

and

$$C(x) = (s_2x^2 + s_1x + s_0)\, Q_3(x)$$

where $s_2, s_1, s_0 \in GF(2^8)$ is a codeword polynomial.

Denote the symbol sequence s_2, s_1, s_0 by $\overline{S}$. Then C(x) corresponds to the IS

$$\underbrace{\overline{S}\ \overline{S} \cdots \overline{S}\ s_2}_{\text{223 symbols}}\ \underbrace{s_1\ s_0\ \overline{S}\ \overline{S} \cdots \overline{S}}_{\text{32 symbols}}$$

which is a codeword. Similarly, $\{R_5\}$ and $\{R_g\}$ have no common elements and g(x) divides

$$Q_5(x) = \frac{x^{255}-1}{x^5-1} = \sum_{j=0}^{50} x^{5j}$$

and

$$C(x) = (s_4x^4 + s_3x^3 + \cdots + s_0)\, Q_5(x)$$

where $s_i \in GF(2^8)$ is a codeword corresponding to an IS. Clearly, ISs of length 15 and 17 are not codewords since $\{R_{15}\}$ and $\{R_{17}\}$ each have elements contained in $\{R_g\}$. (Note that CSs are special cases of ICs.)

C. RELIABILITY TESTING OF A BERLEKAMP RS ENCODER

The 4 nonzero test sequences appearing in Table 10 were contrived by Berlekamp. None are valid codewords. The only nonzero row of information symbols

1	2	3	4	5
z_0 z_7				
00000010	00000100	00001000	00010000	00000000

Table 10. A Realiability Test for a Berlekamp Encoder

Symbol Sequence					
1	2	3	4	5	
z_0 z_7					
00000010	00000100	00001000	00010000	00000000	C_i i>>254
00000000	00000000	00000000	00000000	00000000	
00000000	00000000	00000000	00000000	00000000	(all night run)
...	...	...	...	...	
00000000	00000000	00000000	00000000	00000000	C_{32}
01001001	10010010	00100100	01001000	00000000	
11001110	10011100	00111000	01110001	00000000	
01011100	10111000	01110000	11100000	00000000	
01000101	10001010	00010101	00101010	00000000	
00011110	00111101	01111010	11110100	00000000	
11001111	10011111	00111111	01111110	00000000	
10100110	01001101	10011010	00110100	00000000	
10100111	01001110	10011101	00111011	00000000	
11111111	11111110	11111101	11111010	00000000	
01001110	10011101	00111011	01110110	00000000	
10100011	01000110	10001101	00011010	00000000	
11011011	10110110	01101100	11011001	00000000	
10111011	01110111	11101111	11011111	00000000	
01000001	10000010	00000101	00001011	00000000	
01111011	11110111	11101110	11011100	00000000	
10101100	01011001	10110011	01100111	00000000	
00001011	00010111	00101110	01011100	00000000	
10011110	00111100	01111001	11110011	00000000	
00110000	01100001	11000010	10000100	00000000	
01001011	10010110	00101100	01011000	00000000	
01010000	10100000	01000001	10000010	00000000	
10010100	00101000	01010000	10100000	00000000	
00111001	01110010	11100100	11001000	00000000	
00111100	01111001	11110011	11100110	00000000	
00100110	01001100	10011001	00110011	00000000	
00001000	00010000	00100001	01000011	00000000	
11010000	10100001	01000010	10000101	00000000	
10001111	00011110	00111101	01111010	00000000	
10011110	00111100	01111001	11110011	00000000	
00111011	01110110	11101100	11011000	00000000	
01000011	10000110	00001101	00011011	00000000	
00110000	01100001	11000010	10000100	00000000	C_0

was entered after clearing the encoder. Given

$$z = z_0\, z_1 \cdots z_7$$

then

$$\beta z = z_1\, z_2 \cdots z_8$$

where

$$z_8 = \mathrm{Tr}(\beta^8 z) = z_0 + z_1 + z_3 + z_7.$$

Thus each of the nonzero symbols starting with 2 is β times its predecessor. This row of information symbols followed by rows of all zeros far exceeding 222 rows was entered into the encoder in the message mode over a period of 15 hours at a clock input speed of 1.6 MHz. Each clock time interval is comprised of 4 phases, resulting in an effective 4-kHz internal clock speed. Corresponding to the row just below the row labeled C_{32}, the output was switched to the check mode (after the 1.6-MHz clock was switched to the single-step mode to reach the correct phase by single-stepping the clock and monitoring the encoder's internal signals). Each nonzero sequence of symbols from column 2 through 4 should be β times the symbols of the preceding column. This may be readily verified visually because of the simple relationship between z and βz. The initial row was chosen whereby z_8 for each βz is zero. Any plausible sequence of malfunctions during the run would very likely alter the expected outcome.

VIII. CONCLUSIONS

The IC part count of the (255, 223) RS encoder employing Berlekamp's architecture is 39, of which 9 are RAMs. RAMs are classified as Large-Scale Integrated Circuits (LSIs). The remaining 30 are Small-Scale Integrated Circuits (SSIs). The conventional (255, 223) RS encoder contains 60 ICs of which 24 are LSIs (i.e., RAMs and ROMs). Constraints of power, weight and volume clearly favor Berlekamp's architecture in an IC implementation for spacecraft utilization implementation.

The comparison of Very-Large-Scale Integrated Circuit (VLSI) implementations of conventional and Berlekamp-type RS encoders is considerably less conclusive. In a VLSI implementation, logical elements and connection paths reside on a single chip. Logical elements provide processing and memory, and controlled connection paths provide communication between a processor and memory.

Unlike the case for IC design, complexity is not a function of the number of logical elements but rather of the active chip area they occupy. Patterns with inherent regularity such as those associated with ROMs and RAMs are amenable to VLSI designs where active chip area is at a premium (see Ref. 14). In IC designs, external interconnecting wires between ICs contribute insignificantly to propagation delay in a reasonable layout. By contrast, connection paths in VLSI designs can significantly affect active chip area and propagation delays.

A figure of merit for comparing VLSI implementations is the space-time product. Space is a measure of active chip area, whereas time is a measure of throughput. Parallelism in connection paths (where path-sharing is minimized) increases throughput at the expense of chip area. A VLSI implementation of a conventional RS encoder can match the throughput of a Berlekamp type at the expense of additional active chip area. For spacecraft applications, however, the potential throughput (speed) far exceeds the downlink telemetry rates anticipated for the remainder of this century.

NASA and the European Space Agency (ESA) have formed the NASA/ESA Working Group for Space Data Systems Standardization (NEWG). Telemetry channel coding appears in the "Guidelines for Data Communication Standards" (Ref. 15), which specify the (255, 223) RS code whose mathematical characterization appears in Section V herein. The representation of RS symbols will be in the dual basis (Table 4) in accordance with Berlekamp's architecture. The success of the Berlekamp architecture is reflected in its adoption by NASA/ESA in the guidelines (Ref. 15).

REFERENCES

1. Berlekamp, E.R., Algebraic Coding Theory, McGraw Hill Book Company, N.Y., 1968.
2. MacWilliams, F.J., and Sloane, N.J.A., The Theory of Error-Correcting Codes, North-Holland Publishing Co., Amsterdam, 1977.
3. Forney, G.D., Concatenated Codes, The MIT Press, Cambridge, Mass., 1966.
4. Rice, R.F., Channel Coding and Data Compression System Considerations for Efficient Communication of Planetary Imaging Data, Technical Memorandum 33-695, Jet Propulsion Laboratory, Pasadena, Calif., June 1974.
5. Odenwalder, J.P., "Concatenated Reed-Solomon/Viterbi Channel Coding for Advanced Planetary Missions: Analysis, Simulations, and Tests," Submitted to JPL by the Linkabit Corp., San Diego, Calif., under Contract No. 953866.
6. Peterson, W.W., and Weldon, E.J., Error Correcting Codes, 2nd Edition, The MIT Press, Cambridge, Mass., 1972.
7. Rice, R.F., "Comparative Information Rate Advantages of Alternative Deep Space Communication Systems," Proceedings of the International Conference on Performance of Data Communication Systems and Their Applications, Paris, France, Sept. 1981.
8. Rice, R.F., Hilbert, E.E., Lee, J., and Schultsmyer, A., "Block Adaptive Rate Controlled Image Data Compression," Proceedings of the 1979 National Telecommunication Conference, Washington, D.C., Nov. 1979.
9. Liu, K.Y., The Effects of Receiver Tracking Phase on the Performance of the Concatenated Reed-Solomon/Viterbi Coding System," Publication 81-62, Jet Propulsion Laboratory, Pasadena, Calif., May 1981.
10. Liu, K.Y., and Lee, J., "An Experimental Study of the Concatenated Reed-Solomon/Viterbi Channel Coding System Performance and Its Impact on Space Communication," Proceedings of National Telecommunication Conference, New Orleans, La., Nov. 29-Dec. 3, 1981.
11. Berlekamp, E.R., "Performance Analysis of the Interleaved RS (255, 223) Code," unpublished memorandum, Cyclotomics, Inc., Berkeley, Calif., June 1980.
12. Berlekamp, E.R., "Technical Proposal for a Low-Power Reed-Solomon Encoder/Interleaver Using About 30 CMOS IC's," submitted to JPL by Cyclotomics, Inc., in response to RFP No. BP-6-9007.
13. Lempel, A., "Matrix Factorization Over GF(2) and Trace-Orthogonal Bases of GF(2^n)," SIAM J. Comput., 4., 1975, pps. 175-186.

14. Liu, K.Y., "Architecture for VLSI Design of Reed-Solomon Encoders," IEEE Trans. on Computers, Vol. C-31, No. 2, Feb. 1982.

15. NASA/ESA Working Group for Space Data Systems Standardization (NEWG), "Guidelines for Data Communications Standards: Telemetry Channel Coding — Issue 1," 18 Jan. 1982.