The VLSI Design of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm

T. K. Truong and L. J. Deutsch
Communications Systems Research Section
I. S. Reed, I.-S. Hsu, K. Wang, and C.-S. Yeh
University of Southern California

E. R. Berlekamp has developed for the Jet Propulsion Laboratory a bit-serial multiplication algorithm for the encoding of Reed-Solomon (RS) codes, using a dual basis over a Galois field. The conventional RS-encoder for long codes often requires look-up tables to perform the multiplication of two field elements. Berlekamp's algorithm requires only shifting and exclusive-OR operations. It is shown in this paper that the new dual-basis (255, 223) RS-encoder can be realized readily on a single VLSI chip with NMOS technology.

I. Introduction

A concatenated Reed-Solomon/Viterbi channel encoding system has been suggested both by the European Space Agency (ESA) (Ref. 1) and JPL (Refs. 2, 3) for the deep-space downlink. The standard RS-encoder design developed by JPL assumes the following codes and parameters.

Let \( GF(2^m) \) be a finite field. Then an RS code is a sequence of the symbols in \( GF(2^m) \). This sequence of symbols can be considered to be the coefficients of a polynomial. The code polynomial of such a code is

\[
C(x) = \sum_{i=0}^{d-1} c_i x^i
\]  

(1)

where \( c_i \in GF(2^m) \).

The parameters of an RS code are summarized as follows:

- \( m \) = number of bits per symbol
- \( n = 2^m - 1 \) = the length of a codeword in symbols
- \( t = \) maximum number of error symbols that can be corrected
- \( d = 2t + 1 \) = design distance
- \( 2t \) = number of check symbols
- \( k = n - 2t \) = number of information symbols

In the JPL design, \( m = 8 \), \( n = 255 \), \( t = 16 \), \( d = 33 \), \( 2t = 32 \), and \( k = 223 \). This code is the (255, 223) RS code.
The generator polynomial of an RS code is defined by

\[ g(x) = \sum_{j=b}^{b+2t-1} (x - \gamma^j) = \sum_{i=0}^{2t} g_i x^i \]  

(2)

where \( b \) is a nonnegative integer, usually chosen to be 1, and \( \gamma \) is a primitive element in \( GF(2^m) \). In order to reduce the complexity of the encoder it is desirable to make the coefficients of \( g(x) \) symmetric so that \( g(x) = x^{-d} g(1/x) \). To accomplish this \( b \) must be chosen to satisfy \( 2b + d - 2 = 2^m - 1 \). Thus for the JPL code \( b = 112 \).

Let \( f(x) = c_{2t} x^{2t} + c_{2t-1} x^{2t-1} + \cdots + c_{m-1} x^{m-1} \) and \( P(x) = c_0 + c_1 x + \cdots + c_{2t-1} x^{2t-1} \) be the information polynomial and the check polynomial, respectively. Then the encoded RS code polynomial is represented by

\[ C(x) = f(x) + P(x) \]

(3)

To be an RS code \( C(x) \) must be also a multiple of \( g(x) \). That is,

\[ C(x) = q(x) g(x) \]

(4)

To find \( P(x) \) in Eq. (3) such that Eq. (4) is true, divide \( f(x) \) by \( g(x) \). The division algorithm yields

\[ f(x) = q(x) g(x) + r(x) \]

(5)

Also let \( r(x) = -P(x) \), then by Eq. (5)

\[ q(x) g(x) = f(x) - r(x) = f(x) + P(x) = C(x) \]

(6)

Figure 1 shows the structure of a \( t \)-error correcting RS encoder over \( GF(2^m) \). In Fig. 1 \( R_i \), for \( 0 \leq i \leq 2t - 1 \) and \( Q \) are \( m \)-bit registers. Initially all registers are set to zero, and both switches (controlled by a control signal \( SL \)) are set to position A.

The information symbols \( c_{m-1}, \ldots, c_{2t} \) are fed into the division circuit of the encoder and also transmitted out of the encoder one by one. The quotient coefficients are generated and loaded into \( Q \) register sequentially. The remainder coefficients are computed successively. Immediately after \( c_{2t} \) is fed to the circuit, both switches are set to position B. At the very same moment \( c_{2t-1} \) is computed and transmitted. Simultaneously, \( c_t \) is being computed and loaded into register \( R_i \), for \( 0 \leq i \leq 2t - 2 \). Next \( c_{2t-2}, \ldots, c_0 \) are transmitted out of the encoder one by one. \( c_{2t-1}, \ldots, c_0 \) retain their values because the content of \( Q \) is set to zero when the upper switch is at position B.

The complexity of the design of an RS encoder results from the computation of products \( z_i g_i \) for \( 0 \leq i \leq 2t - 2 \). These computations can be performed in several ways (Ref. 3). Unfortunately none of them is suited to the pipeline processing structures usually seen in VLSI design. Recently, Berlekamp (Ref. 4) developed a bit-serial multiplier algorithm that has the features needed to solve this problem. Perlman and Lee (Ref. 5) show in detail the mathematical basis for this algorithm. In this paper Berlekamp's method is applied to the VLSI design of a \( (255, 22^2) \) RS-encoder, which can be implemented on a single VLSI chip.

II. Berlekamp's Bit-Serial Multiplier Algorithm Over \( GF(2^m) \)

In order to understand Berlekamp's multiplier algorithm some mathematical preliminaries are needed. Toward this end the mathematical concepts of the trace and a complementary (or dual) basis are introduced. For more details and proofs see Refs. 3, 4 and 5.

**Definition 1:** The trace of an element \( \beta \) belonging to \( GF(p^m) \), the Galois field of \( p^m \) elements, is defined as follows:

\[ Tr(\beta) = \sum_{k=0}^{m-1} \beta^k \]

In particular, for \( p = 2 \),

\[ Tr(\beta) = - \sum_{k=0}^{m-1} (\beta^2)^k \]

The trace has the following properties:

1. \( [Tr(\beta)]^p = \beta + \beta^p + \cdots + \beta^{p-1} = Tr(\beta) \), where \( \beta \in GF(p^m) \). This implies that \( Tr(\beta) \in GF(p) \); i.e., the trace is on the ground field \( GF(p) \).

2. \( Tr(\beta + \gamma) = Tr(\beta) + Tr(\gamma) \), where \( \beta, \gamma \in GF(p^m) \)

3. \( Tr(c\beta) = cTr(\beta) \), where \( c \in GF(p) \).

4. \( Tr(1) \equiv m(\text{mod } p) \).

**Definition 2:** A basis \( \{\mu_i\} \) in \( GF(p^m) \) is a set of \( m \) linearly independent elements in \( GF(p^m) \).

**Definition 3:** Two bases \( \{\mu_i\} \) and \( \{\lambda_j\} \) are said to be complementary or the dual of one another if

\[ Tr(\mu_i, \lambda_j) = \begin{cases} 1, & i = k \\ 0, & j \neq k \end{cases} \]
The basis \( \{ \mu_k \} \) is called the original basis, and the basis \( \{ \lambda_k \} \) is called the dual basis.

**Lemma:** If \( \alpha \) is a root of an irreducible polynomial of degree \( m \) in \( GF(p^m) \), then \( \{ \alpha^k \} \) for \( 0 \leq k < m - 1 \) is a basis of \( GF(p^m) \). The basis \( \{ \alpha^k \} \) for \( 0 \leq k < m - 1 \) is called the normal or natural basis of \( GF(p^m) \).

**Theorem 1** (Theorem 19 in Ref. 4): Every basis has a complementary basis.

**Corollary 1:** Suppose the bases \( \{ \mu_k \} \) and \( \{ \lambda_k \} \) are complementary. Then a field element \( z \) can be expressed in the dual basis \( \{ \lambda_k \} \) by the expansion

\[
z = \sum_{k=0}^{m-1} z_k \lambda_k = \sum_{k=0}^{m-1} Tr(z\mu_k)\lambda_k
\]

where \( z_k = Tr(z\mu_k) \) is the \( k \)th coefficient of the dual basis.

**Proof:** Let \( z = z_0 \lambda_0 + z_1 \lambda_1 + \cdots + z_{m-1} \lambda_{m-1} \). Multiply both sides by \( \alpha^k \) and take the trace. Then by Def. 3 and the properties of the trace,

\[
Tr(z\alpha^k) = Tr\left( \sum_{i=0}^{m-1} z_i \lambda_i \mu_k \right) = z_k
\]

Q.E.D.

The following corollary is an immediate consequence of Corollary 1.

**Corollary 2:** The product \( w = zv \) of two field elements in \( GF(p^m) \) can be expressed in the dual basis by the expansion

\[
w = \sum_{k=0}^{m-1} Tr(zv\mu_k)\lambda_k
\]

where \( Tr(zv\mu_k) \) is the \( k \)th coefficient of the dual basis for the product of two field elements.

These two corollaries provide a theoretical basis for the new RS-encoder algorithm.

**III. A Simple Example of Berlekamp's Algorithm Applied to an RS-Encoder**

This section follows the treatment in Ref. 3. It is included here for two purposes. First, Ref. 3 is not readily available for most readers. Second, this example is included to illustrate how Berlekamp's new bit-serial multiplier algorithm can be used to realize the RS-encoder structure presented in Fig. 1.

Consider a \((15, 11)\) RS code: over \( GF(2^4) \). For this code, \( m = 4, n = 15, t = 2, d = 2t + 1 = 5, \) and \( n - 2t = 11 \) information symbols. Let \( \alpha \) be a root of the primitive irreducible polynomial \( f(x) = x^5 + x + 1 \) over \( GF(2) \). \( \alpha \) satisfies \( \alpha^{11} = 1 \). An element \( z \) in \( GF(2^4) \) is representable by \( 0 \) or \( \alpha \) for some \( k \). 0 \( \leq k \) \( \leq 14 \). \( z \) can be represented also by a polynomial in \( \alpha \) over \( GF(2) \). This is the representation of \( GF(2^4) \) in the normal basis \( \{ \alpha^k \} \) for \( 0 \leq k \leq 3 \). That is, \( z = u_0 + u_1 \alpha + u_2 \alpha^2 + u_3 \alpha^3 \), where \( u_k \in GF(2) \) for \( 0 \leq k \leq 3 \).

In Table 1, the first column is the index or logarithm of an element in base \( \alpha \). The logarithm of the zero element is denoted by an asterisk. Columns 2 to 5 show the 4-tuples of the coefficients of the elements expressed as polynomials.

The trace of an element \( z \) in \( GF(2^4) \) is found by Def. 1 and the properties of the trace to be

\[
TR(z) = u_0 Tr(1) + u_1 Tr(\alpha) + u_2 Tr(\alpha^2) + u_3 Tr(\alpha^3)
\]

where \( Tr(1) \equiv 4 (\mod 2) = 0, Tr(\alpha) = Tr(\alpha^2) = \alpha + \alpha^2 + \alpha^4 + \alpha^8 = 0 \) and \( Tr(\alpha^3) = \alpha^2 + \alpha^6 + \alpha^{12} = 1 \). Thus \( TR(z) = u_3 \).

The trace element \( \alpha^k \) in \( GF(2^4) \) is listed in column 3 of Table 1.

By Def. 2 any set of four linearly independent elements can be used as a basis for the field \( GF(2^4) \). To find the dual basis of the normal basis \( \{ \alpha^k \} \) in \( GF(2^4) \), let a field element \( z \) be expressed in dual basis \( \{ \lambda_0, \lambda_1, \lambda_2, \lambda_3 \} \). From Corollary 1 the coefficients of \( z \) are \( z_k = Tr(z\alpha^k) \) for \( 0 \leq k \leq 3 \). Thus \( z_0 = Tr(z), z_1 = Tr(z\alpha), z_2 = Tr(z\alpha^2) \) and \( z_3 = Tr(z\alpha^3) \). Let \( z = \alpha^i \) for some \( i, 0 \leq i \leq 14 \). Thus a coefficient \( z_k \), for \( 0 \leq k \leq 3 \), of an element \( z \) in the dual space can be obtained by cyclically shifting the trace column in Table 1 upward by \( k \) positions where the first row is excluded. These appropriately shifted columns of coefficients are shown in Table 1 as the last four columns. In Table 1 the elements of the dual basis, \( \lambda_0, \lambda_1, \lambda_2, \lambda_3 \), are underlined. Evidently \( \lambda_0 = \alpha^{14}, \lambda_1 = \alpha^2, \lambda_2 = a, \lambda_3 = 1 \) are the four elements of the dual basis.

In order to make the generator polynomial \( g(x) \) symmetric \( b \) must satisfy the equation \( 2b + d - 2 = 2^m - 1 \). Thus \( b = 6 \) for this code. The \( \gamma \) in Eq. (2) can be any primitive element in \( GF(2^m) \). It will be shown in Section IV that \( \gamma \) can be chosen to simplify the binary mapping matrix. In this example let \( \gamma = \alpha \).

Thus the generator polynomial is given by

\[
g(x) = \prod_{j=0}^{9} (x - \alpha^j) = \sum_{i=0}^{9} g_i x^i
\] (7)
where \( g_0 = g_4 = 1, g_1 = g_3 = \alpha^3 \) and \( g_2 = \alpha \).

Let \( g \) be expressed in the normal basis \( \{1, \alpha, \alpha^2, \alpha^3\} \). Let \( z \) be a field element, be expressed in the dual basis; i.e., \( z = z_0 \lambda_0 + z_1 \lambda_1 + z_2 \lambda_2 + z_3 \lambda_3 \). In Fig. 1 the products \( zg_i \) for \( 0 \leq i \leq 3 \) needs to be computed.

Since \( g_3 = g_1 \), it is necessary to compute only \( zg_0, zg_2 \), and \( zg_3 \). Let the products \( zg_i \) for \( 0 \leq i \leq 2 \) be represented in the dual basis. By Corollary 2 \( zg_i \) can be expressed in the dual basis as

\[
\begin{bmatrix}
  z \\
  g_0 \\
  g_1 \\
  g_2 \\
\end{bmatrix} = \sum_{k=0}^{3} T_k(z) \lambda_k
\]

where \( T_k(z) = \text{Tr}(zg_i \alpha^k) \) is the \( k \)th coefficient (or \( k \)th bit) of \( zg_i \) for \( 0 \leq i \leq 2 \) and \( 0 \leq k \leq 3 \).

The present problem is to express \( T^{(k)}(z) \) recursively in terms of \( T^{(k-1)}(z) \) for \( 1 \leq k \leq 3 \). Initially for \( k = 0 \),

\[
\begin{bmatrix}
  T_0^{(0)}(z) \\
  T_1^{(0)}(z) \\
  T_2^{(0)}(z) \\
\end{bmatrix} = \begin{bmatrix}
  \text{Tr}(zg_0) \\
  \text{Tr}(zg_1) \\
  \text{Tr}(zg_2) \\
\end{bmatrix} = \begin{bmatrix}
  \text{Tr}(z^0) \\
  \text{Tr}(z^1) \\
  \text{Tr}(z^3) \\
\end{bmatrix} = \begin{bmatrix}
  z_0 \\
  z_1 \\
  z_3 \\
\end{bmatrix}
\]

where \( \text{Tr}(z^j) \) is \( j \)th root of \( \alpha \) for \( 0 \leq j \leq 3 \). Equation (9) can be expressed in a matrix form as follows:

\[
\begin{bmatrix}
  T_0^{(0)}(z) \\
  T_1^{(0)}(z) \\
  T_2^{(0)}(z) \\
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  0 & 1 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
  z_0 \\
  z_1 \\
  z_3 \\
\end{bmatrix}
\]

The above matrix is the \( 3 \times 4 \) binary mapping matrix of the problem.

To compute \( T^{(k)}(z) \) for \( k > 0 \), observe that \( T^{(k)}(z) = \text{Tr}((zg_i \alpha^{k-1}). \) Hence \( T^{(k)}(z) \) is obtained from \( T^{(k-1)}(z) \) by replacing \( z \) by \( y = \text{Tr}(z^m) \). Then \( T^{(0)}_k(z) = \text{Tr}(zg_i \alpha^k) \) for \( 0 \leq k \leq 3 \). Then \( T^{(0)}_k(z) \) is obtained from \( T^{(k-1)}(z) \) by replacing \( z \) by \( y = \text{Tr}(z^m) \) for \( 0 \leq k \leq 3 \).

To recapitulate \( zg_i = T^{(0)}_0(z) + T^{(1)}_1(z) + T^{(2)}_2(z) + T^{(3)}_3(z) \), where \( 0 \leq i \leq 3 \) and \( z = z_0 \lambda_0 + z_1 \lambda_1 + z_2 \lambda_2 + z_3 \lambda_3 \), can be computed by Berlekamp's bit-serial multiplier algorithm as follows:

1. Initially for \( k = 0 \), compute \( T_0^{(0)}(z) \), \( T_1^{(0)}(z) \) and \( T_2^{(0)}(z) \) by Eq. (10). Also \( T_3^{(0)}(z) = T_0^{(1)}(z) \).
2. For \( k = 1, 2, 3 \), compute \( T_k^{(1)}(z) \) by

\[
T_k^{(1)}(z) = T^{(k-1)}_k(y)
\]

where \( 0 \leq k \leq 3 \) and \( y = \text{Tr}(z^m) \) for \( 0 \leq m \leq 3 \). Equation (10) can be expressed in a matrix form as follows:

\[
\begin{bmatrix}
  T_0^{(0)}(z) \\
  T_1^{(0)}(z) \\
  T_2^{(0)}(z) \\
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  0 & 1 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
  z_0 \\
  z_1 \\
  z_3 \\
\end{bmatrix}
\]

The above matrix is the \( 3 \times 4 \) binary mapping matrix of the problem.

\[
\text{IV. A VLSI Architecture of a (255, 223) RS-Encoder with Dual-Basis Multiplier}
\]

In this section an architecture is designed to implement (255, 223) RS-encoder using Berlekamp's multiplier algorithm. The circuit is a direct mapping from an encoder using Berlekamp's bit-serial algorithm as developed in the previous sections to an architectural design. This architecture can be realized quite readily on a single NMOS VLSI chip.

Let \( GF(2^8) \) be generated by \( \alpha \), where \( \alpha \) is a root of a primitive irreducible polynomial \( f(x) = x^8 + x^7 + x^6 + x^4 + 1 \) over \( GF(2) \). The normal basis of this field is \( \{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7\} \). The representations of this field in both the normal basis and its dual basis are tabulated in Appendix A. From Corollary 1 the coefficients of a field element \( \alpha \) can be obtained from \( z_k = \text{Tr}(\alpha^k \lambda_0) \) for \( 0 \leq k \leq 7 \), where \( \lambda_0 = \text{Tr}(x^m) \).

From Table A-1 in Appendix A, the dual basis
The normal basis is the ordered set \((\alpha^{0}, \alpha^{10}, \alpha^{202}, \alpha^{203}, \alpha^{204}, \alpha^{205}, \alpha^{206})\).

It was mentioned previously that \(\gamma\) in Eq. (2) can be chosen to simplify the binary mapping matrix. Two binary matrices, one for the primitive element \(\gamma = \alpha^{11}\) and the other for \(\gamma = \alpha\), were computed. It was found that the binary mapping matrix for \(\gamma = \alpha^{11}\) had a smaller number of 1's. Hence this binary mapping matrix was used in the design. For this case the generator polynomial \(g(x)\) of the RS-encoder over \(GF(2^8)\) was given by

\[
g(x) = \prod_{i=0}^{143} (x - \alpha^{11/i}) + \sum_{i=0}^{32} g_i x^i
\]

where \(g_0 = g_{32} = 1\), \(g_1 = g_{31} = \alpha^{249}\), \(g_2 = g_{30} = \alpha^{59}\), \(g_3 = g_{29} = \alpha^{66}\), \(g_4 = g_{28} = \alpha^4\), \(g_5 = g_{27} = \alpha^{43}\), \(g_6 = g_{26} = \alpha^{126}\), \(g_7 = g_{25} = \alpha^{251}\), \(g_8 = g_{24} = \alpha^{97}\), \(g_9 = g_{23} = \alpha^{30}\), \(g_{10} = g_{22} = \alpha^3\), \(g_{11} = g_{21} = \alpha^{57}\), \(g_{12} = g_{20} = \alpha^{50}\), \(g_{13} = g_{19} = \alpha^{64}\), \(g_{14} = g_{18} = \alpha^{170}\), \(g_{15} = g_{17} = \alpha^6\), and \(g_{16} = \alpha^{24}\).

The binary mapping matrix for the coefficients of the generator polynomial in Eq. (11) is computed and shown in Appendix B. The feedback term \(T_f\) in Berlekamp's algorithm is found in this case to be:

\[
T_f = Tr(\alpha^8 z) \cdot Tr((\alpha^7 + \alpha^2 + \alpha + 1)z) = z_0 + z_1 + z_2 + z_7
\]

In the following a VLSI chip architecture is designed to realize a (255, 223) RS-encoder using the above parameters and Berlekamp's algorithms. An overall block diagram of this chip is shown in Fig. 2. In Fig. 2 VDD and GND are power pins. CLK is a clock signal, which is in general a periodic square wave. The information symbols are fed into the chip from the data-in pin DIN bit-by-bit. Similarly the encoded codeword is transmitted out of the chip from the data-out pin DOUT sequentially. The control signal LM load mode is set to 0 (logic 0) when the information symbols are loaded into the chip. Otherwise, LM is set to 0.

The input data and LM signals are synchronized by the CLK signal, while the operations of the circuit and output data signal are synchronized by two nonoverlapping clock signals. To save space, dynamic registers are used in this circuit. The contents of a 1-bit dynamic register with reset is shown in Fig. 4. The delay of DOUT sequential. The control signal LM, DIN, and the input and output flip-flops.

Figure 5 shows the block diagram of a (255, 223) RS-encoder over \(GF(2^8)\) using Berlekamp's bit-serial multiplier algorithm. The circuit is divided into five units. The circuits in each unit are discussed in the following:

1. Product Unit: The Product Unit is used to compute \(T_p, T_{31}, \ldots, T_0\). This circuit is realized by a Programmable Logic Array (PLA) circuit [6]. Since \(T_0 = T_{31}, T_1 = T_{30}, \ldots, T_{15} = T_{17}\), only \(T_0, T_{31}, \ldots, T_{17}\) and \(T_{16}\) are actually implemented in the PLA circuit. \(T_0, \ldots, T_{15}\) are connected directly to \(T_{31}, \ldots, T_{17}\), respectively. Over other circuits a PLA circuit has the advantage of being easy to reconfigure.

2. Remainder Unit: The Remainder Unit is used to store the coefficients of the remainder during the division process. In Fig. 5, \(S_i\) for \(0 < i < 30\) are 8-bit shift registers with reset. The addition in the circuit is a modulo 2 addition or Exclusive-OR operation. While \(c_{32}\) is being fed to the circuit, \(c_{31}\) is being computed and transmitted sequentially from the circuit. Simultaneously \(c_i\) is computed and then loaded into \(S_i\) for \(0 < i < 30\). Then \(c_{30}, \ldots, c_0\) are transmitted out of the encoder bit-by-bit.

3. Quotient Unit: In Fig. 5, \(Q\) and \(R\) represent a 7-bit shift register with reset and an 8-bit shift register with reset and parallel load, respectively. \(R\) and \(Q\) store the currently operating coefficients and the next coefficient of the quotient polynomial, respectively. A logic diagram of register \(R\) is shown in Fig. 6. It is loaded into \(R\) every eight clock cycles, where \(0 < i < 7\). Immediately after all 223 information symbols are fed into the circuit, the control signal SL changes to logic 0. Henceforth the contents of \(Q\) and \(R\) are zero so that the check symbols in the Remainder Unit sustain their values.

4. I/O Unit: This unit handles the input/output operations. In Fig. 5 both \(F_6\) and \(F_1\) are flip-flops. A pass transistor controlled by \(\phi_1\) is inserted before \(F_1\) for the purpose of synchronization. Control signal SL selects whether a bit of an information symbol or a check symbol is to be transmitted.

5. Control Unit: The Control Unit generates the necessary control signals. This unit is further divided into portions, as shown in Fig. 7. The two-phase clock generator circuit in Ref. 6 is used to convert a clock signal into two nonoverlapping clock signals \(\phi_1\) and \(\phi_2\). In Fig. 8 is shown a logic diagram of the circuit for generating control signals START and SL. Control signal START resets all registers and the divide-by-8
counter before the encoding process begins. Control signal LD is simply generated by a divide-by-8 counter to load the $r_i$'s into the $R_i$'s in parallel.

Since a codeword contains 255 symbols, the computation of a complete encoded codeword requires 255 "symbol cycles." A symbol cycle is the time interval for executing a complete cycle of Berlekamp's algorithm. Since a symbol has 8 bits, a symbol cycle contains 8 "bit cycles." A bit cycle is the time interval for executing one step in Berlekamp's algorithm. In this design a bit cycle requires a period of the clock cycle.

The layout design of this (255, 223) RS-encoder is shown in Fig. 9. Before the design of the layout each circuit was simulated on a general-purpose computer by using SPICE (a transistor-level circuit simulation program) (Ref. 7). The circuit requires about 3000 transistors, while a similar JPL design requires 30 CMOS IC chips (Ref. 5). This RS-encoder design will be fabricated and tested in the near future.

V. Concluding Remarks

A VLSI structure is developed for a Reed-Solomon encoder using Berlekamp's bit-serial multiplier algorithm. This structure is both regular and simple.

The circuit in Fig. 2 can be modified easily to encode an RS code with a different field representation and different parameters other than those used in Section IV. Table 2 shows the primary modifications needed in the circuit to change a given parameter.

Acknowledgment

The authors wish to acknowledge Marvin Perlman of the Information Systems Engineering Section of JPL. His initial work with E. Berlekamp eventually resulted in the acceptance of the Berlekamp code as a guideline for both NASA and ESA. Also, his tutorial paper on the algorithm (Reference 5) was particularly helpful to the authors.

References

### Table 1. Representations of elements over GF(2^3) generated by α^3 = α + 1

<table>
<thead>
<tr>
<th>Power *</th>
<th>Elements in normal base</th>
<th>( \text{Tr}(\alpha^i) )</th>
<th>Elements in dual base</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0010</td>
<td>0</td>
<td>0010 ( \alpha^2 )</td>
</tr>
<tr>
<td>2</td>
<td>0100</td>
<td>0</td>
<td>0100 ( \alpha^3 )</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>1</td>
<td>1001</td>
</tr>
<tr>
<td>4</td>
<td>0011</td>
<td>0</td>
<td>0011</td>
</tr>
<tr>
<td>5</td>
<td>0110</td>
<td>0</td>
<td>0110</td>
</tr>
<tr>
<td>6</td>
<td>1100</td>
<td>1</td>
<td>1101</td>
</tr>
<tr>
<td>7</td>
<td>1011</td>
<td>1</td>
<td>1011</td>
</tr>
<tr>
<td>8</td>
<td>0101</td>
<td>0</td>
<td>0101</td>
</tr>
<tr>
<td>9</td>
<td>1010</td>
<td>1</td>
<td>1011</td>
</tr>
<tr>
<td>10</td>
<td>0111</td>
<td>0</td>
<td>0111</td>
</tr>
<tr>
<td>11</td>
<td>1110</td>
<td>1</td>
<td>1111</td>
</tr>
<tr>
<td>12</td>
<td>1111</td>
<td>1</td>
<td>1110</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>1</td>
<td>1100</td>
</tr>
<tr>
<td>14</td>
<td>1001</td>
<td>1</td>
<td>1000 ( \alpha^3 )</td>
</tr>
</tbody>
</table>

### Table 2. The primary modifications of the encoder circuit in Fig. 2 needed to change a parameter

<table>
<thead>
<tr>
<th>Parameter to be changed</th>
<th>The value used for the circuit in Fig. 2</th>
<th>New value</th>
<th>The circuits of Fig. 2 that require modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator polynomial</td>
<td>eq. (8) ( g(x) )</td>
<td>( g(x) )</td>
<td>The PLA of the Product Unit needs to be changed</td>
</tr>
<tr>
<td>Finite field used</td>
<td>( GF(2^8) ) ( GF(2^m) )</td>
<td>( GF(2^m) )</td>
<td>All registers are ( m )-bit registers, except ( Q ) is a ( (m - 1) )-bit register. A divide-by-( m ) counter is used. (The generator polynomial may not be changed.)</td>
</tr>
<tr>
<td>Error-correcting capability</td>
<td>16 ( t )</td>
<td>( t )</td>
<td>2t-2 shift registers are required in the Remainder Unit. (The generator polynomial is also changed.)</td>
</tr>
<tr>
<td>Number of information symbols</td>
<td>223 ( k )</td>
<td>( k )</td>
<td>None is changed, since ( k ) is implicitly contained in the control signal LM</td>
</tr>
</tbody>
</table>
Fig. 1. A structure of a 1-error correcting RS-encoder

Fig. 2. Symbolic diagram of a RS encoder chip

Fig. 3. Logic diagram of a 1-bit dynamic register with reset

Fig. 4. The signals of DIN, LM, CLK, φ₁, φ₂, START, and DOUT
Fig. 5. Block diagram of the encoder

Fig. 6. Block diagram of register R

Fig. 7. Block diagram of the Control Unit

Fig. 8. Logic diagram of the circuit for generating control signals START and SL

R_i: A 1-BIT REGISTER WITH RESET
Fig. 9. Layout of the (255, 223) RS-encoder chip
Appendix A

In this appendix all 256 elements in $GF(2^8)$ are listed in Table A-1. These field elements are expressed in both the normal basis and its dual basis.

<table>
<thead>
<tr>
<th>Power $j$</th>
<th>Elements in normal base</th>
<th>$\text{Tr}(\alpha^j)$</th>
<th>Elements in dual base</th>
<th>Power $j$</th>
<th>Elements in normal base</th>
<th>$\text{Tr}(\alpha^j)$</th>
<th>Elements in dual base</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000000000</td>
<td>0</td>
<td>00000000</td>
<td>35</td>
<td>111010000</td>
<td>0</td>
<td>00110111</td>
</tr>
<tr>
<td>0</td>
<td>000000000</td>
<td>0</td>
<td>01111111</td>
<td>36</td>
<td>010101111</td>
<td>0</td>
<td>01101110</td>
</tr>
<tr>
<td>1</td>
<td>000000101</td>
<td>1</td>
<td>01111111</td>
<td>37</td>
<td>101011111</td>
<td>1</td>
<td>11011100</td>
</tr>
<tr>
<td>2</td>
<td>000001000</td>
<td>1</td>
<td>11111110</td>
<td>38</td>
<td>110110100</td>
<td>0</td>
<td>10110000</td>
</tr>
<tr>
<td>3</td>
<td>000010000</td>
<td>1</td>
<td>11111101</td>
<td>39</td>
<td>001100011</td>
<td>0</td>
<td>01110000</td>
</tr>
<tr>
<td>4</td>
<td>000100000</td>
<td>1</td>
<td>11111010</td>
<td>40</td>
<td>010000110</td>
<td>1</td>
<td>11100000</td>
</tr>
<tr>
<td>5</td>
<td>001000000</td>
<td>1</td>
<td>11110101</td>
<td>41</td>
<td>110001000</td>
<td>1</td>
<td>10000001</td>
</tr>
<tr>
<td>6</td>
<td>010000000</td>
<td>1</td>
<td>11101010</td>
<td>42</td>
<td>000111111</td>
<td>1</td>
<td>10000001</td>
</tr>
<tr>
<td>7</td>
<td>100000000</td>
<td>1</td>
<td>11010101</td>
<td>43</td>
<td>000111110</td>
<td>0</td>
<td>00000110</td>
</tr>
<tr>
<td>8</td>
<td>100011111</td>
<td>1</td>
<td>11010111</td>
<td>44</td>
<td>001111100</td>
<td>1</td>
<td>00001100</td>
</tr>
<tr>
<td>9</td>
<td>100100101</td>
<td>0</td>
<td>01010111</td>
<td>45</td>
<td>011100000</td>
<td>0</td>
<td>00011000</td>
</tr>
<tr>
<td>10</td>
<td>101010101</td>
<td>1</td>
<td>10011110</td>
<td>46</td>
<td>011011000</td>
<td>0</td>
<td>00111100</td>
</tr>
<tr>
<td>11</td>
<td>101101101</td>
<td>0</td>
<td>01101110</td>
<td>47</td>
<td>010011111</td>
<td>0</td>
<td>01100001</td>
</tr>
<tr>
<td>12</td>
<td>110111011</td>
<td>1</td>
<td>11101001</td>
<td>48</td>
<td>110011110</td>
<td>1</td>
<td>11000001</td>
</tr>
<tr>
<td>13</td>
<td>001111101</td>
<td>0</td>
<td>01100011</td>
<td>49</td>
<td>001101111</td>
<td>1</td>
<td>00001110</td>
</tr>
<tr>
<td>14</td>
<td>011110101</td>
<td>1</td>
<td>11010011</td>
<td>50</td>
<td>001100110</td>
<td>1</td>
<td>01100011</td>
</tr>
<tr>
<td>15</td>
<td>111101001</td>
<td>1</td>
<td>01000111</td>
<td>51</td>
<td>011001110</td>
<td>0</td>
<td>00011100</td>
</tr>
<tr>
<td>16</td>
<td>011011111</td>
<td>0</td>
<td>00111101</td>
<td>52</td>
<td>011011000</td>
<td>0</td>
<td>01110000</td>
</tr>
<tr>
<td>17</td>
<td>110111100</td>
<td>0</td>
<td>00111001</td>
<td>53</td>
<td>011001111</td>
<td>0</td>
<td>01100001</td>
</tr>
<tr>
<td>18</td>
<td>001110111</td>
<td>0</td>
<td>01101010</td>
<td>54</td>
<td>011001110</td>
<td>1</td>
<td>01100001</td>
</tr>
<tr>
<td>19</td>
<td>011011110</td>
<td>1</td>
<td>11011010</td>
<td>55</td>
<td>110111000</td>
<td>1</td>
<td>11000110</td>
</tr>
<tr>
<td>20</td>
<td>111011000</td>
<td>1</td>
<td>11001101</td>
<td>56</td>
<td>001111111</td>
<td>1</td>
<td>10001100</td>
</tr>
<tr>
<td>21</td>
<td>010111111</td>
<td>1</td>
<td>10011001</td>
<td>57</td>
<td>011111100</td>
<td>0</td>
<td>00011100</td>
</tr>
<tr>
<td>22</td>
<td>101111110</td>
<td>0</td>
<td>00100110</td>
<td>58</td>
<td>111111000</td>
<td>0</td>
<td>00110111</td>
</tr>
<tr>
<td>23</td>
<td>111111010</td>
<td>0</td>
<td>01001110</td>
<td>59</td>
<td>011111111</td>
<td>0</td>
<td>01100010</td>
</tr>
<tr>
<td>24</td>
<td>011000101</td>
<td>1</td>
<td>10011010</td>
<td>60</td>
<td>111111110</td>
<td>1</td>
<td>11001100</td>
</tr>
<tr>
<td>25</td>
<td>111000010</td>
<td>0</td>
<td>00111010</td>
<td>61</td>
<td>011101000</td>
<td>1</td>
<td>10011000</td>
</tr>
<tr>
<td>26</td>
<td>010000111</td>
<td>0</td>
<td>01101010</td>
<td>62</td>
<td>111101000</td>
<td>0</td>
<td>00111000</td>
</tr>
<tr>
<td>27</td>
<td>100010110</td>
<td>1</td>
<td>11010100</td>
<td>63</td>
<td>011010110</td>
<td>1</td>
<td>01100000</td>
</tr>
<tr>
<td>28</td>
<td>100010111</td>
<td>1</td>
<td>11010000</td>
<td>64</td>
<td>111001110</td>
<td>1</td>
<td>11001100</td>
</tr>
<tr>
<td>29</td>
<td>100100011</td>
<td>0</td>
<td>01000011</td>
<td>65</td>
<td>001011111</td>
<td>1</td>
<td>10001100</td>
</tr>
<tr>
<td>30</td>
<td>101001011</td>
<td>1</td>
<td>10110001</td>
<td>66</td>
<td>010101110</td>
<td>0</td>
<td>00011001</td>
</tr>
<tr>
<td>31</td>
<td>110011101</td>
<td>0</td>
<td>01000011</td>
<td>67</td>
<td>101011000</td>
<td>1</td>
<td>00100011</td>
</tr>
<tr>
<td>32</td>
<td>001111101</td>
<td>1</td>
<td>10000110</td>
<td>68</td>
<td>110111111</td>
<td>0</td>
<td>01001110</td>
</tr>
<tr>
<td>33</td>
<td>001111011</td>
<td>0</td>
<td>00001110</td>
<td>69</td>
<td>001110100</td>
<td>1</td>
<td>10001101</td>
</tr>
<tr>
<td>34</td>
<td>011101000</td>
<td>0</td>
<td>00011011</td>
<td>70</td>
<td>011100100</td>
<td>0</td>
<td>00011010</td>
</tr>
<tr>
<td>Power /</td>
<td>Elements in normal base</td>
<td>Tr(α')</td>
<td>Elements in dual base</td>
<td>Power /</td>
<td>Elements in normal base</td>
<td>Tr(α')</td>
<td>Elements in dual base</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------</td>
<td>--------</td>
<td>----------------------</td>
<td>---------</td>
<td>-------------------------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td>71</td>
<td>11100100</td>
<td>0</td>
<td>00110100</td>
<td>112</td>
<td>01000111</td>
<td>1</td>
<td>10010100</td>
</tr>
<tr>
<td>72</td>
<td>0.001111</td>
<td>0</td>
<td>01101001</td>
<td>113</td>
<td>1'001110</td>
<td>0</td>
<td>0010101</td>
</tr>
<tr>
<td>73</td>
<td>10011011</td>
<td>1</td>
<td>11100111</td>
<td>114</td>
<td>10011011</td>
<td>0</td>
<td>01010010</td>
</tr>
<tr>
<td>74</td>
<td>10111011</td>
<td>1</td>
<td>10100111</td>
<td>115</td>
<td>10110001</td>
<td>1</td>
<td>10100101</td>
</tr>
<tr>
<td>75</td>
<td>11100001</td>
<td>0</td>
<td>01001111</td>
<td>116</td>
<td>11100101</td>
<td>0</td>
<td>01001011</td>
</tr>
<tr>
<td>76</td>
<td>00100011</td>
<td>1</td>
<td>10011110</td>
<td>117</td>
<td>00101101</td>
<td>1</td>
<td>10010101</td>
</tr>
<tr>
<td>77</td>
<td>11000101</td>
<td>0</td>
<td>00111101</td>
<td>118</td>
<td>01010100</td>
<td>0</td>
<td>00101101</td>
</tr>
<tr>
<td>78</td>
<td>00100011</td>
<td>0</td>
<td>01110101</td>
<td>119</td>
<td>10110011</td>
<td>1</td>
<td>01011010</td>
</tr>
<tr>
<td>79</td>
<td>00100011,0</td>
<td>1</td>
<td>11100100</td>
<td>120</td>
<td>11100001</td>
<td>1</td>
<td>10101011</td>
</tr>
<tr>
<td>80</td>
<td>01001010</td>
<td>1</td>
<td>11010100</td>
<td>121</td>
<td>01000101</td>
<td>0</td>
<td>01101011</td>
</tr>
<tr>
<td>81</td>
<td>10011010</td>
<td>1</td>
<td>11010010</td>
<td>122</td>
<td>10001010</td>
<td>1</td>
<td>11010111</td>
</tr>
<tr>
<td>82</td>
<td>10110111</td>
<td>1</td>
<td>10100100</td>
<td>123</td>
<td>10010111</td>
<td>1</td>
<td>10101111</td>
</tr>
<tr>
<td>83</td>
<td>11100011</td>
<td>0</td>
<td>01000100</td>
<td>124</td>
<td>10100001</td>
<td>0</td>
<td>01110111</td>
</tr>
<tr>
<td>84</td>
<td>01010101</td>
<td>1</td>
<td>10010010</td>
<td>125</td>
<td>11000101</td>
<td>1</td>
<td>10111110</td>
</tr>
<tr>
<td>85</td>
<td>10101010</td>
<td>0</td>
<td>00100010</td>
<td>126</td>
<td>00001011</td>
<td>0</td>
<td>11110100</td>
</tr>
<tr>
<td>86</td>
<td>01000100</td>
<td>0</td>
<td>01000101</td>
<td>127</td>
<td>00010100</td>
<td>1</td>
<td>11111000</td>
</tr>
<tr>
<td>87</td>
<td>01000001</td>
<td>1</td>
<td>10001010</td>
<td>128</td>
<td>00110100</td>
<td>1</td>
<td>11110001</td>
</tr>
<tr>
<td>88</td>
<td>01000010</td>
<td>0</td>
<td>00010101</td>
<td>129</td>
<td>01110000</td>
<td>1</td>
<td>11100100</td>
</tr>
<tr>
<td>89</td>
<td>01000010</td>
<td>0</td>
<td>00101010</td>
<td>130</td>
<td>11010000</td>
<td>1</td>
<td>11001010</td>
</tr>
<tr>
<td>90</td>
<td>10001111</td>
<td>0</td>
<td>01010110</td>
<td>131</td>
<td>00100111</td>
<td>1</td>
<td>10001111</td>
</tr>
<tr>
<td>91</td>
<td>10001111,1</td>
<td>1</td>
<td>10100011</td>
<td>132</td>
<td>00011011</td>
<td>0</td>
<td>00011011</td>
</tr>
<tr>
<td>92</td>
<td>10110101</td>
<td>0</td>
<td>01101111</td>
<td>133</td>
<td>10011001</td>
<td>0</td>
<td>01010110</td>
</tr>
<tr>
<td>93</td>
<td>11101011</td>
<td>1</td>
<td>11101100</td>
<td>134</td>
<td>11111111</td>
<td>0</td>
<td>01101101</td>
</tr>
<tr>
<td>94</td>
<td>01011010</td>
<td>0</td>
<td>01101100</td>
<td>135</td>
<td>11111101</td>
<td>1</td>
<td>01110010</td>
</tr>
<tr>
<td>95</td>
<td>01110101</td>
<td>1</td>
<td>11010010</td>
<td>136</td>
<td>01101011</td>
<td>0</td>
<td>01110010</td>
</tr>
<tr>
<td>96</td>
<td>11110001</td>
<td>1</td>
<td>11010000</td>
<td>137</td>
<td>11010101</td>
<td>1</td>
<td>11001000</td>
</tr>
<tr>
<td>97</td>
<td>01100001</td>
<td>0</td>
<td>01100000</td>
<td>138</td>
<td>01010011</td>
<td>1</td>
<td>10100000</td>
</tr>
<tr>
<td>98</td>
<td>11100001</td>
<td>1</td>
<td>11000000</td>
<td>139</td>
<td>10100110</td>
<td>0</td>
<td>00100001</td>
</tr>
<tr>
<td>99</td>
<td>00000001</td>
<td>1</td>
<td>10000001</td>
<td>140</td>
<td>11000111</td>
<td>0</td>
<td>01000010</td>
</tr>
<tr>
<td>100</td>
<td>00000011</td>
<td>0</td>
<td>00000001</td>
<td>141</td>
<td>00100011</td>
<td>1</td>
<td>10000111</td>
</tr>
<tr>
<td>101</td>
<td>00000100</td>
<td>0</td>
<td>00000011</td>
<td>142</td>
<td>00100010</td>
<td>0</td>
<td>00010010</td>
</tr>
<tr>
<td>102</td>
<td>00001000</td>
<td>0</td>
<td>00001111</td>
<td>143</td>
<td>01000100</td>
<td>0</td>
<td>00020010</td>
</tr>
<tr>
<td>103</td>
<td>00110000</td>
<td>0</td>
<td>00011111</td>
<td>144</td>
<td>10100010</td>
<td>0</td>
<td>00100010</td>
</tr>
<tr>
<td>104</td>
<td>01100000</td>
<td>0</td>
<td>01101111</td>
<td>145</td>
<td>10011011</td>
<td>0</td>
<td>01001001</td>
</tr>
<tr>
<td>105</td>
<td>11000000</td>
<td>0</td>
<td>01111111</td>
<td>146</td>
<td>10100101</td>
<td>1</td>
<td>10100010</td>
</tr>
<tr>
<td>106</td>
<td>00000101</td>
<td>1</td>
<td>01111110</td>
<td>147</td>
<td>11010011</td>
<td>0</td>
<td>01000010</td>
</tr>
<tr>
<td>107</td>
<td>00110110</td>
<td>1</td>
<td>11111001</td>
<td>148</td>
<td>00101101</td>
<td>1</td>
<td>10000101</td>
</tr>
<tr>
<td>108</td>
<td>00011100</td>
<td>1</td>
<td>11110010</td>
<td>149</td>
<td>01011010</td>
<td>0</td>
<td>00010010</td>
</tr>
<tr>
<td>109</td>
<td>00111000</td>
<td>1</td>
<td>11110010</td>
<td>150</td>
<td>10100100</td>
<td>0</td>
<td>00100100</td>
</tr>
<tr>
<td>110</td>
<td>01100001</td>
<td>1</td>
<td>11100101</td>
<td>151</td>
<td>11111111</td>
<td>0</td>
<td>01110100</td>
</tr>
<tr>
<td>111</td>
<td>11100000</td>
<td>1</td>
<td>11001010</td>
<td>152</td>
<td>01011001</td>
<td>1</td>
<td>10000101</td>
</tr>
</tbody>
</table>

Table A-1 (contd)

Original Page is of Poor Quality
<table>
<thead>
<tr>
<th>Power $i$</th>
<th>Elements in normal base</th>
<th>$Tr(\delta')$</th>
<th>Elements in dual base</th>
<th>Power $i$</th>
<th>Elements in normal base</th>
<th>$Tr(\delta')$</th>
<th>Elements in dual base</th>
</tr>
</thead>
<tbody>
<tr>
<td>153</td>
<td>10110010</td>
<td>0</td>
<td>00100101</td>
<td>194</td>
<td>01001001</td>
<td>0</td>
<td>01101000</td>
</tr>
<tr>
<td>154</td>
<td>11100011</td>
<td>0</td>
<td>01010100</td>
<td>195</td>
<td>10100101</td>
<td>1</td>
<td>11010000</td>
</tr>
<tr>
<td>155</td>
<td>01101001</td>
<td>1</td>
<td>10010101</td>
<td>196</td>
<td>10100101</td>
<td>1</td>
<td>10100000</td>
</tr>
<tr>
<td>156</td>
<td>01100010</td>
<td>0</td>
<td>01001010</td>
<td>197</td>
<td>11000001</td>
<td>0</td>
<td>01000000</td>
</tr>
<tr>
<td>157</td>
<td>10000001</td>
<td>1</td>
<td>10101010</td>
<td>198</td>
<td>00001001</td>
<td>1</td>
<td>10000000</td>
</tr>
<tr>
<td>158</td>
<td>10001001</td>
<td>0</td>
<td>01010100</td>
<td>200</td>
<td>00010001</td>
<td>0</td>
<td>00000010</td>
</tr>
<tr>
<td>159</td>
<td>10000101</td>
<td>0</td>
<td>01010100</td>
<td>201</td>
<td>00100000</td>
<td>0</td>
<td>00000010</td>
</tr>
<tr>
<td>160</td>
<td>10011001</td>
<td>1</td>
<td>10101001</td>
<td>202</td>
<td>01001000</td>
<td>0</td>
<td>00001000</td>
</tr>
<tr>
<td>161</td>
<td>10110101</td>
<td>0</td>
<td>01001110</td>
<td>203</td>
<td>10100000</td>
<td>0</td>
<td>00001000</td>
</tr>
<tr>
<td>162</td>
<td>10111101</td>
<td>1</td>
<td>10010111</td>
<td>204</td>
<td>11000111</td>
<td>0</td>
<td>01000001</td>
</tr>
<tr>
<td>163</td>
<td>11111101</td>
<td>0</td>
<td>01001100</td>
<td>205</td>
<td>00001001</td>
<td>1</td>
<td>10000010</td>
</tr>
<tr>
<td>164</td>
<td>01111101</td>
<td>1</td>
<td>10010101</td>
<td>206</td>
<td>01000010</td>
<td>0</td>
<td>00000101</td>
</tr>
<tr>
<td>165</td>
<td>11111010</td>
<td>0</td>
<td>00101000</td>
<td>207</td>
<td>00100010</td>
<td>0</td>
<td>00000101</td>
</tr>
<tr>
<td>166</td>
<td>01110001</td>
<td>1</td>
<td>10110001</td>
<td>208</td>
<td>01001010</td>
<td>0</td>
<td>00011011</td>
</tr>
<tr>
<td>167</td>
<td>11110011</td>
<td>1</td>
<td>11011010</td>
<td>209</td>
<td>10100000</td>
<td>0</td>
<td>00011011</td>
</tr>
<tr>
<td>168</td>
<td>01001011</td>
<td>1</td>
<td>10011011</td>
<td>210</td>
<td>11001101</td>
<td>0</td>
<td>01011110</td>
</tr>
<tr>
<td>169</td>
<td>10010110</td>
<td>0</td>
<td>01010110</td>
<td>211</td>
<td>10110101</td>
<td>0</td>
<td>00101110</td>
</tr>
<tr>
<td>170</td>
<td>10101110</td>
<td>0</td>
<td>01011101</td>
<td>212</td>
<td>01001010</td>
<td>0</td>
<td>00011011</td>
</tr>
<tr>
<td>171</td>
<td>11100001</td>
<td>1</td>
<td>10101100</td>
<td>213</td>
<td>01001100</td>
<td>0</td>
<td>00111011</td>
</tr>
<tr>
<td>172</td>
<td>00101011</td>
<td>0</td>
<td>01110100</td>
<td>214</td>
<td>01011000</td>
<td>1</td>
<td>10111110</td>
</tr>
<tr>
<td>173</td>
<td>01001010</td>
<td>1</td>
<td>11101000</td>
<td>215</td>
<td>01101010</td>
<td>1</td>
<td>10111110</td>
</tr>
<tr>
<td>174</td>
<td>10010100</td>
<td>0</td>
<td>00111001</td>
<td>216</td>
<td>11001101</td>
<td>1</td>
<td>10111110</td>
</tr>
<tr>
<td>175</td>
<td>10110111</td>
<td>1</td>
<td>11010001</td>
<td>217</td>
<td>11011101</td>
<td>1</td>
<td>10111110</td>
</tr>
<tr>
<td>176</td>
<td>11100011</td>
<td>1</td>
<td>00101001</td>
<td>218</td>
<td>01001000</td>
<td>1</td>
<td>11101111</td>
</tr>
<tr>
<td>177</td>
<td>00110011</td>
<td>1</td>
<td>10011110</td>
<td>219</td>
<td>01101000</td>
<td>1</td>
<td>11101111</td>
</tr>
<tr>
<td>178</td>
<td>01101010</td>
<td>0</td>
<td>00011101</td>
<td>220</td>
<td>01101101</td>
<td>1</td>
<td>10111100</td>
</tr>
<tr>
<td>179</td>
<td>11101000</td>
<td>0</td>
<td>01111011</td>
<td>221</td>
<td>10101000</td>
<td>0</td>
<td>01111000</td>
</tr>
<tr>
<td>180</td>
<td>00101111</td>
<td>0</td>
<td>01111011</td>
<td>222</td>
<td>00101100</td>
<td>0</td>
<td>00001001</td>
</tr>
<tr>
<td>181</td>
<td>01011110</td>
<td>1</td>
<td>11101000</td>
<td>223</td>
<td>01001000</td>
<td>1</td>
<td>11000001</td>
</tr>
<tr>
<td>182</td>
<td>10111000</td>
<td>1</td>
<td>11011001</td>
<td>224</td>
<td>10100000</td>
<td>1</td>
<td>10000001</td>
</tr>
<tr>
<td>183</td>
<td>11111111</td>
<td>1</td>
<td>10100111</td>
<td>225</td>
<td>00101101</td>
<td>1</td>
<td>10000100</td>
</tr>
<tr>
<td>184</td>
<td>01111001</td>
<td>0</td>
<td>01100111</td>
<td>226</td>
<td>00101110</td>
<td>0</td>
<td>00001001</td>
</tr>
<tr>
<td>185</td>
<td>11100011</td>
<td>1</td>
<td>10011111</td>
<td>227</td>
<td>01011000</td>
<td>0</td>
<td>00001001</td>
</tr>
<tr>
<td>186</td>
<td>01100011</td>
<td>1</td>
<td>10011111</td>
<td>228</td>
<td>01011000</td>
<td>0</td>
<td>00001001</td>
</tr>
<tr>
<td>187</td>
<td>11000110</td>
<td>0</td>
<td>00111110</td>
<td>229</td>
<td>11101101</td>
<td>0</td>
<td>01001110</td>
</tr>
<tr>
<td>188</td>
<td>00010111</td>
<td>0</td>
<td>01111011</td>
<td>230</td>
<td>01101001</td>
<td>1</td>
<td>10001110</td>
</tr>
<tr>
<td>189</td>
<td>00010110</td>
<td>1</td>
<td>11111101</td>
<td>231</td>
<td>11010010</td>
<td>0</td>
<td>00111010</td>
</tr>
<tr>
<td>190</td>
<td>00111100</td>
<td>1</td>
<td>11111111</td>
<td>232</td>
<td>01000011</td>
<td>0</td>
<td>01111010</td>
</tr>
<tr>
<td>191</td>
<td>01011000</td>
<td>1</td>
<td>11110110</td>
<td>233</td>
<td>01000110</td>
<td>1</td>
<td>11101011</td>
</tr>
<tr>
<td>192</td>
<td>10110000</td>
<td>1</td>
<td>11110110</td>
<td>234</td>
<td>10001100</td>
<td>1</td>
<td>11101110</td>
</tr>
<tr>
<td>193</td>
<td>11100111</td>
<td>1</td>
<td>10110100</td>
<td>235</td>
<td>10001100</td>
<td>1</td>
<td>11101110</td>
</tr>
</tbody>
</table>
Table A-1 (contd)

<table>
<thead>
<tr>
<th>Power $j$</th>
<th>Elements in normal base</th>
<th>Tr($a'$)</th>
<th>Power $j$</th>
<th>Elements in normal base</th>
<th>Tr($a'$)</th>
<th>Elements in dual base</th>
<th>Elements in dual base</th>
</tr>
</thead>
<tbody>
<tr>
<td>235</td>
<td>10011111</td>
<td>1</td>
<td>245</td>
<td>01111100</td>
<td>1</td>
<td>11100110</td>
<td>11100110</td>
</tr>
<tr>
<td>236</td>
<td>10111001</td>
<td>0</td>
<td>246</td>
<td>11111000</td>
<td>1</td>
<td>11001101</td>
<td>11001101</td>
</tr>
<tr>
<td>237</td>
<td>11110101</td>
<td>1</td>
<td>247</td>
<td>01110111</td>
<td>1</td>
<td>10011011</td>
<td>10011011</td>
</tr>
<tr>
<td>238</td>
<td>01101101</td>
<td>0</td>
<td>248</td>
<td>11101110</td>
<td>0</td>
<td>00110110</td>
<td>00110110</td>
</tr>
<tr>
<td>239</td>
<td>11011010</td>
<td>1</td>
<td>249</td>
<td>01011011</td>
<td>0</td>
<td>01101101</td>
<td>01101101</td>
</tr>
<tr>
<td>240</td>
<td>00110011</td>
<td>1</td>
<td>250</td>
<td>10110110</td>
<td>1</td>
<td>11011011</td>
<td>11011011</td>
</tr>
<tr>
<td>241</td>
<td>01100110</td>
<td>0</td>
<td>251</td>
<td>11101110</td>
<td>1</td>
<td>10101111</td>
<td>10101111</td>
</tr>
<tr>
<td>242</td>
<td>11001100</td>
<td>0</td>
<td>252</td>
<td>01010001</td>
<td>0</td>
<td>01101111</td>
<td>01101111</td>
</tr>
<tr>
<td>243</td>
<td>00011111</td>
<td>0</td>
<td>253</td>
<td>10100010</td>
<td>1</td>
<td>11011111</td>
<td>11011111</td>
</tr>
<tr>
<td>244</td>
<td>00111110</td>
<td>1</td>
<td>254</td>
<td>11000111</td>
<td>1</td>
<td>10111111</td>
<td>10111111</td>
</tr>
</tbody>
</table>
Appendix B

The binary mapping matrix for $\gamma = a^{11}$ of the (255, 223) RS-encoder is given by

$$
\begin{bmatrix}
T_0 \\
T_1 \\
T_2 \\
T_3 \\
T_4 \\
T_5 \\
T_6 \\
T_7 \\
T_8 \\
T_9 \\
T_{10} \\
T_{11} \\
T_{12} \\
T_{13} \\
T_{14} \\
T_{15} \\
T_{16}
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
$$