General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

JPL PUBLICATION 81-87

Architecture for VLSI Design of Reed-Solomon Encoders

K. Y. Liu

June 8, 1981

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 81-87

Architecture for VLSI Design of Reed-Solomon Encoders

K. Y. Liu

June 8, 1981

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.

ACKNOWLEDGMENT

This document consolidates one phase of research under the NASA End-to-End Information System (NEEDS) program and the LSI Product Assurance Technology program sponsored by the Chief Engineer's Office at NASA. The author expresses his gratitude to Dr. A. J. Ferrari, J. A. Hunter, Richard B. Miller, and R. H. Nixon for their support which made this publication possible. Also, the author thanks Dr. J. J. Lee, M. Perlman, and R. F. Rice for their comments and suggestions. Finally, the author thanks J. T. Sumida and J. C. Packard for providing the detailed logic design and fabrication, respectively.

ABSTRACT

In this document, the logic structure of a universal VLSI chip called the symbol-slice Reed-Solomon (RS) encoder chip is presented. An RS encoder can be constructed by cascading and properly interconnecting a group of such VLSI chips. As a design example, it is shown that a (255,223) RS encoder requiring around 40 discrete CMOS IC's may be replaced by an RS encoder consisting of four identical interconnected VLSI RS encoder chips. Besides the size advantage, the VLSI RS encoder also has the potential advantages of requiring less power and having a higher reliability.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION I

INTRODUCTION

Reed-Solomon (RS) codes (Ref. 1) are nonbinary BCH codes. These codes can correct both random and burst errors over a communication channel. Recently concatenated coding systems using RS codes as the outer codes have been proposed for space communication to achieve very low error probabilities (Refs. 2 to 7). Several deep space flight projects such as the Voyager at Uranus encounter, the Galileo, and the International Solar Polar Mission (ISPM) have also considered using the concatenated RS/Viterbi channel coding scheme. Hence RS codes are quite important for space communications.

The complexity of an RS encoder is proportional to the error-correcting capability of the code, the speed of the encoding, and the interleaving level used (Ref. 4). For reliable space communication there is a need to use RS codes with large error-correcting capability and large interleaving level (Refs. 4, 5, 8, and 9. Hence one is especially interested in minimizing the complexity of RS encoders for space communication applications. In a spacecraft the power, size, and reliability requirements are usually quite severe. Thus there is considerable interest in a VLSI (Very Large Scale Integration) RS encoder which has the potential for significant savings in size, weight, and power while at the same time providing higher reliability over an RS encoder implemented in discrete logic circuits.

This document introduces a symbol-sliced logic structure suitable for a VLSI implementation of RS encoders. By cascading and properly interconnecting a group of such VLSI chips, each consisting of a fixed portion of the encoder, it is possible to obtain an RS encoder with any desired error-correcting capability

and interleaving level. As a design example, it is shown that a (255,223) RS encoder requiring 40 discrete CMOS IC's may be replaced by an RS encoder consisting of four identical interconnected VLSI encoder chips. It is also shown that these VLSI RS encoder chips can be paralleled to improve the encoding speed.

SECTION II

BASIC CONCEPTS OF FINITE FIELDS

A field is a set of elements, including 0 and 1, any pair of which may be added or multiplied (denoted by + and *) to give a unique result in the field. The addition and multiplication are associative and commutative, and multiplication distributes over addition in the usual way, i.e.,

$$u * (v+w) = u*v+u*w$$

Every field element u has a unique negative element -u in the same field such that

$$u + (-u) = 0$$

Every nonzero field element u has a unique reciprocal field element 1/u, such that

$$u * (1/u) = 1$$

For every field element u

$$0 + u = u = 1 * u$$

and

$$0 * u = 0$$

If the number of elements in a field is infinite, then it is called an infinite field. Examples of infinite fields are:

(a) The rational number field.

(b) The real number field.

(c) The complex number field.

If the number of elements in a field is finite, then it is called a finite field or a Galios field GF(q), where q is the number of elements in the field. Two examples of finite fields are given as follows. The first example is the finite field GF(p) which is formed by integers modulo p, where p is a prime. If p = 2, then the field is called GF(2). GF(2) contains only two elements, i.e., 0 and 1. The addition and multiplication tables of GF(2) are given as below:

	0	1
0	0	1
1	1	0

Addition

	0	1
0	0	0
1	0	1

Multiplication

The second example is the finite field formed by polynomials modulo and irreducible polynomial of degree m with coefficients in GF(p), where p is a prime. The definition of an irreducible polynomial is given as follows. A polynomial

$$p(x) = \sum_{i=0}^{m} a_i \, x^i$$

with $a_i \in GF(p)$ is called irreducible over GF(p) if there exists no polynomials A(x) and B(x) with coefficients from GF(p) such that

$$p(x) = A(x) \, B(x)$$

where

$$1 \leq \text{Degree of } A(x) \leq m-1$$

As an example, if $p = 2$, then the polynomials

$$x^8 + x^4 + x^3 + x^2 + 1$$

and

$$x^4 + x + 1$$

are irreducible over GF(2).

Now the multiplicative structure of finite fields will be discussed. If a field contains an element α, then the least positive integer N for which $\alpha^N = 1$ is called the order of α. In a finite field of q elements, GF(q), there is a primitive element α, i.e., an element of order $q - 1$. Every nonzero element of GF(q) can be expressed as a power of α.

Next the vector space structure of finite fields will be presented. To do it, one needs the following definition of a primitive polynomial. An irreducible polynomial of degree m over GF(q) is called primitive if it has a primitive element of $GF(q^m)$ as a root. A finite field of q^m elements, $GF(q^m)$, can be considered as an m-dimensional vector space over GF(q). A choice for a basis of $GF(q^m)$ over GF(q) is the set

$$\left\{1, \alpha, \alpha^2, ---, \alpha^{m-1}\right\}$$

which is called the canonical basis, where α is a root of a primitive polynomial of degree m over GF(q). In vector form

$$\left\{1, \alpha, \alpha^2, ---, \alpha^{m-1}\right\}$$

is represented as

$$1 \leftrightarrow (0, 0, ---, 0, 0, 1)$$

$$\alpha \leftrightarrow (0, 0, ---, 0, 1, 0)$$

$$\alpha^2 \leftrightarrow (0, 0, ---, 1, 0, 0)$$

$$\vdots$$

$$\alpha^{m-1} \leftrightarrow (1, 0, ---, 0, 0, 0)$$

whereas 0 is mapped to (0, 0, ---, 0). Thus all elements in $GF(q^m)$ except 0 can be formed by linear combinations of the canonical basis

$$1, \alpha, \alpha^2, ---, \alpha^{m-1}$$

As an example, the Galois field of 2^4 elements, $GF(2^4)$, may be formed as the field of polynomials over GF(2) modules $(x^4 + x + 1)$, which is a primitive polynomial of degree 4. Let α be a root of $x^4 + x + 1$, i.e.,

$$\alpha^4 + \alpha + 1 = 0$$

Then by the definition of the primitive polynomial, all elements of $GF(2^4)$ except 0 are powers of α. The representation of α^i for $4 \leq i \leq 15$ can be determined by the primitive polynomial. In this example α is a root of $x^4 + x + 1$ over GF(2). Thus

$$\alpha^4 + \alpha + 1 = 0$$

It follows that

$$\alpha^4 = -\alpha - 1$$

Since $-1 = 1$ in GF(2), one has

$$\alpha^4 = \alpha + 1$$

The rest of the element α^i for $5 \leq i \leq 15$ can be obtained likewise. The 15 nonzero field elements of $GF(2^4)$ are shown below in both multiplicative and vector space forms.

$$
\begin{array}{lll}
\alpha^0 = 1 & = & (0001) \\
\alpha^1 = \alpha & = & (0010) \\
\alpha^2 = \alpha^2 & = & (0100) \\
\alpha^3 = \alpha^3 & = & (1000) \\
\alpha^4 = \alpha + 1 & = & (0011) \\
\alpha^5 = \alpha^2 + \alpha & = & (0110) \\
\alpha^6 = \alpha^3 + \alpha^2 & = & (1100) \\
\alpha^7 = \alpha^3 + \alpha + 1 & = & (1011) \\
\alpha^8 = \alpha^2 + 1 & = & (0101) \\
\alpha^9 = \alpha^3 + \alpha & = & (1010) \\
\alpha^{10} = \alpha^2 + \alpha + 1 & = & (0111) \\
\alpha^{11} = \alpha^3 + \alpha^2 + \alpha & = & (1110) \\
\alpha^{12} = \alpha^3 + \alpha^2 + \alpha + 1 & = & (1111) \\
\alpha^{13} = \alpha^3 + \alpha^2 + 1 & = & (1101) \\
\alpha^{14} = \alpha^3 + 1 & = & (1001) \\
\alpha^{15} = 1 & = & (0001) = \alpha^0
\end{array}
$$

Another example is the finite field $GF(2^8)$ generated by the primitive polynomial

$$x^8 + x^4 + x^3 + x^2 + 1$$

The exponents versus elements and elements versus exponents tables for this $GF(2^8)$ are shown in Table 1 and Table 2, respectively.

Table 1. Exponents versus elements in a finite field $GF(2^8)$ generated by the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT
0	1	1	2	2	4	126	102	127	204	128	133	255	1
3	8	4	16	5	32	129	23	130	46	131	92		
6	64	7	128	8	29	132	184	133	109	134	218		
9	58	10	116	11	232	135	169	136	79	137	158		
12	205	13	135	14	19	138	33	139	66	140	132		
15	38	16	76	17	152	141	21	142	42	143	84		
18	45	19	90	20	180	144	168	145	77	146	154		
21	117	22	234	23	201	147	41	148	82	149	164		
24	143	25	3	26	6	150	85	151	170	152	73		
27	12	28	24	29	48	153	146	154	57	155	114		
30	96	31	192	32	157	156	228	157	213	158	193		
33	39	34	78	35	156	159	115	160	230	161	209		
36	37	37	74	38	148	162	191	163	99	164	198		
39	53	40	106	41	212	165	145	166	63	167	126		
42	181	43	119	44	238	168	252	169	229	170	215		
45	193	46	159	47	35	171	179	172	123	173	246		
48	70	49	140	50	5	174	241	175	255	176	227		
51	10	52	20	53	40	177	219	178	171	179	75		
54	80	55	160	56	93	180	150	181	49	182	98		
57	186	58	105	59	210	183	196	184	149	185	55		
60	185	61	111	62	222	186	110	187	220	188	165		
63	161	64	95	65	190	189	87	190	174	191	65		
66	97	67	194	68	153	192	130	193	25	194	50		
69	47	70	94	71	188	195	100	196	200	197	141		
72	101	73	202	74	137	198	7	199	14	200	28		
75	15	76	30	77	60	201	56	202	112	203	224		
78	120	79	240	80	253	204	221	205	167	206	83		
81	231	82	211	83	187	207	166	208	81	209	162		
84	107	85	214	86	177	210	89	211	178	212	121		
87	127	88	254	89	225	213	242	214	249	215	239		
90	223	91	163	92	91	216	195	217	155	218	43		
93	182	94	113	95	226	219	86	220	172	221	69		
96	217	97	175	98	67	222	138	223	9	224	18		
99	134	100	17	101	34	225	36	226	72	227	144		
102	68	103	136	104	13	228	61	229	122	230	244		
105	26	106	52	107	104	231	245	232	247	233	243		
108	208	109	189	110	103	234	251	235	235	236	203		
111	206	112	129	113	31	237	139	238	11	239	22		
114	62	115	124	116	248	240	44	241	88	242	176		
117	237	118	199	119	147	243	125	244	250	245	233		
120	59	121	118	122	236	246	207	247	131	248	27		
123	197	124	151	125	51	249	54	250	108	251	216		
						252	173	253	71	254	142		

Table 2. Elements versus exponents in a finite field $GF(2^8)$ generated by the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT
1	0	2	1	3	25	127	87	128	7	129	112
4	2	5	50	6	26	130	192	131	247	132	140
7	198	8	3	9	223	133	128	134	99	135	13
10	51	11	238	12	27	136	103	137	74	138	222
13	104	14	199	15	75	139	237	140	49	141	197
16	4	17	100	18	224	142	254	143	24	144	227
19	14	20	52	21	141	145	165	146	153	147	119
22	239	23	129	24	28	148	38	149	184	150	180
25	193	26	105	27	248	151	124	152	17	153	68
28	200	29	8	30	76	154	146	155	217	156	35
31	113	32	5	33	138	157	32	158	137	159	46
34	101	35	47	36	225	160	55	161	63	162	209
37	36	38	15	39	33	163	91	164	149	165	188
40	53	41	147	42	142	166	207	167	205	168	144
43	218	44	240	45	18	169	135	170	151	171	178
46	130	47	69	48	29	172	220	173	252	174	190
49	181	50	194	51	125	175	97	176	242	177	86
52	106	53	39	54	249	178	211	179	171	180	20
55	185	56	201	57	154	181	42	182	93	183	158
58	9	59	120	60	77	184	132	185	60	186	57
61	228	62	114	63	166	187	83	188	71	189	109
64	6	65	191	66	139	190	65	191	162	192	31
67	98	68	102	69	221	193	45	194	67	195	216
70	48	71	253	72	226	196	183	197	123	198	164
73	152	74	37	75	179	199	118	200	196	201	23
76	16	77	145	78	34	202	73	203	236	204	127
79	136	80	54	81	208	205	12	206	111	207	246
82	148	83	206	84	143	208	108	209	161	210	59
85	150	86	219	87	189	211	82	212	41	213	157
88	241	89	210	90	19	214	85	215	170	216	251
91	92	92	131	93	56	217	96	218	134	219	177
94	70	95	64	96	30	220	187	221	204	222	62
97	66	98	182	99	163	223	90	224	203	225	89
100	195	101	72	102	126	226	95	227	176	228	156
103	110	104	107	105	58	229	169	230	160	231	81
106	40	107	84	108	250	232	11	233	245	234	22
109	133	110	186	111	61	235	235	236	122	237	117
112	202	113	94	114	155	238	44	239	215	240	79
115	159	116	10	117	21	241	174	242	213	243	233
118	121	119	43	120	78	244	230	245	231	246	173
121	212	122	229	123	172	247	232	248	116	249	214
124	115	125	243	126	167	250	244	251	234	252	168
						253	80	254	88	255	175

The addition of two field elements in $GF(2^n)$ is performed by component-wise addition in the vector representations of the two elements. For example in $GF(2^8)$, $\alpha^{26} + \alpha^{238}$ is given by

$$\begin{aligned} 6 &= (0,0,0,0,0,1,1,0) \quad \leftrightarrow \quad \alpha^{26} \\ +\ 11 &= (0,0,0,0,1,0,1,1) \quad \leftrightarrow \quad \alpha^{238} \\ \hline 13 &= (0,0,0,0,1,1,0,1) \quad \leftrightarrow \quad \alpha^{104} \end{aligned}$$

The multiplication of two field elements in $GF(2^n)$ is performed by a modulo (2^n-1) addition on the exponents of the two elements. For example in $GF(2^8)$, $\alpha^{26} * \alpha^{238}$ is given by

$$\begin{aligned} 6 &= (0,0,0,0,0,1,1,0) \quad \leftrightarrow \quad \alpha^{26} \\ *)\ 11 &= (0,0,0,0,1,0,1,1) \quad \leftrightarrow \quad \alpha^{238} \\ \hline 58 &= (0,0,1,1,1,0,1,0) \quad \leftrightarrow \quad \alpha^{(26\ +\ 238)\ \mathrm{MOD}\ 255} \\ &\qquad\qquad\qquad\qquad\quad = \alpha^{9} \end{aligned}$$

SECTION III

REED-SOLOMON ENCODING PROCEDURES

An RS code word has (2^J-1) symbols, where each symbol has J bits. Of the (2^J-1) symbols there are (2^J-1-2E) information symbols and 2E parity-check symbols, where E is the number of symbols an RS code is able to correct. If one treats the (2^J-1-2E) information symbols as the coefficients of the polynomial

$$f(x) = x^{2E}\left(s_{2^J-1-2E} + s_{2^J-2-2E}\,x + \cdots + s_2 x^{2^J-2-2E} + s_1 x^{2^J-1-2E}\right)$$

where s_i is the ith transmitted symbol, then the 2E parity-check symbols can be obtained as the coefficients of the remainder of

$$f(x)/g(x)$$

where g(x) is the generator polynomial (Ref. 9) of the code. Usually g(x) is defined as

$$g(x) = \prod_{i=1}^{2E} (x-\alpha^i) = \sum_{j=0}^{2E} g_j\, x^j$$

where α is a primitive element of the Galois field $GF(2^J)$, and g_j's are the coefficients of g(x) with $g_{2E} = 1$. The generator polynomial defined above does not have symmetrical coefficients, i.e.,

$$g_j \neq g_{2E-j} \text{ for } j = 0, 1, 2, \cdots, 2E.$$

A block diagram of an RS encoder which generates the remainder of f(x)/g(x) is given in Fig. 1. The switches in Fig. 1 are normally in the "ON" position until the last information symbol gets into the encoder. At this

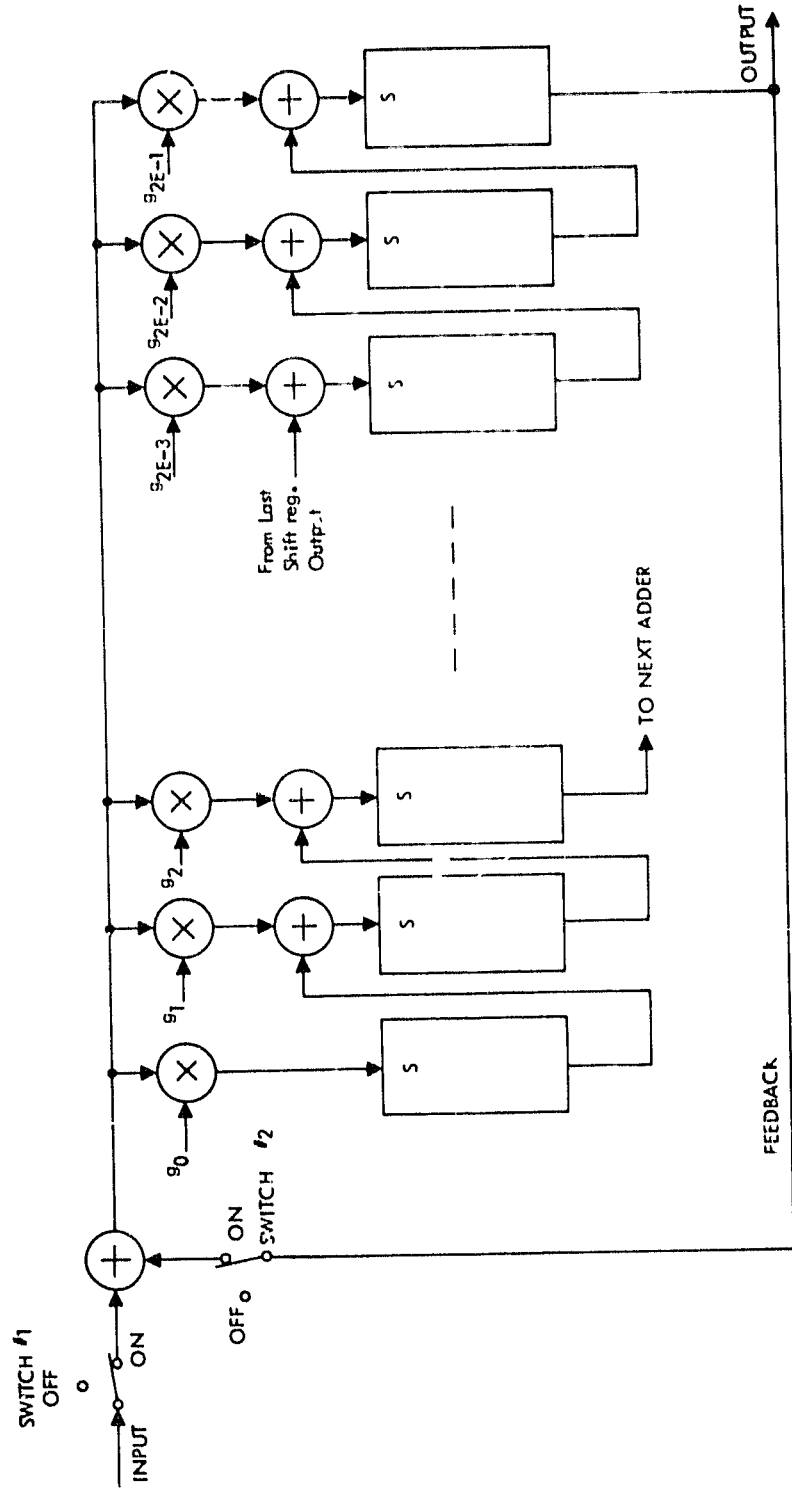


Figure 1. A Block Diagram of a $(2^J-1, 2^J-1-2E)$ RS Encoder Using a Conventional Generator Polynomial.

Legend: ⊕ = Finite Field Adder, ⊗ = Finite Field Multiplier,

S = J-bit Shift Register.

moment all switches are switched to the "OFF" position and the encoder is behaving like a long shift register. The output of the encoder is then taken from the output of the last shift register. Note that in Fig. 1, 2E multipliers are needed in the encoder.

To reduce the number of multipliers needed, a special class of the generator polynomial which has symmetrical coefficients was proposed by Berlekamp (Ref. 10). This generator polynomial is defined as

$$g(x) = \prod_{i=2^{J-1}-E}^{2^{J-1}+E-1} (x-\alpha^i) = \sum_{j=o}^{2E} g_j x^j$$

where

$$g_j = g_{2E-j} \text{ and } g_o = g_{2E} = 1.$$

Note that since $g_o = 1$, only E multipliers are needed (see Fig. 2). Thus using this new generator polynominal will reduce the number of multipliers required by one-half. As an example, the coefficients of all generator polynomials of the form

$$g(x) = \prod_{i=112}^{143} (x - \alpha^{ij})$$

for a 16-error-correcting RS code with 8-bit per symbol are shown in Table 3 for α = 2, 128, 232, 135, 201, 90, 74, 119, respectively.

There are several schemes for interleaving the RS codes (Ref. 5). One scheme illustrated in Fig. 3 as "Interleave B" requires memory only for the parity-check symbols in the encoder is described as follows. In this scheme the input bits are grouped into J-bit symbols and transmitted in their natural order. However every I^{th} symbol belongs to the same code word, where I is the interleaving

Table 3. Generator Polynomial Coefficients for $g(x) = \prod_{i=112}^{143} (x-\alpha^i)$

DEG(x)	α=2	α=128	α=232	α=135	α=201	α=90	α=74	α=119
0	1	1	1	1	1	1	1	1
1	236	227	213	174	139	169	8	73
2	244	151	184	158	11	247	135	192
3	220	109	106	204	140	73	138	158
4	133	47	132	153	63	177	187	246
5	238	104	36	89	43	80	65	160
6	137	26	144	44	223	53	130	94
7	201	184	194	91	97	59	123	159
8	7	213	9	154	236	129	198	171
9	141	4	59	158	41	106	44	5
10	11	133	67	188	88	103	102	240
11	226	174	81	234	156	18	117	129
12	34	238	56	191	102	47	77	124
13	252	97	158	72	145	184	138	188
14	209	157	204	226	98	64	214	242
15	22	232	74	59	31	83	106	173
16	78	6	135	78	220	168	125	80
17	22	232	74	59	31	83	106	173
18	209	157	204	226	98	64	214	242
19	252	97	158	72	145	184	138	188
20	34	238	56	191	102	47	77	124
21	226	174	81	234	156	18	117	129
22	11	133	67	188	88	103	102	240
23	141	4	59	158	41	106	44	5
24	7	213	9	154	236	129	198	171
25	201	184	194	91	97	59	123	159
26	137	26	144	44	223	53	130	94
27	238	104	36	89	43	80	65	160
28	133	47	132	153	63	177	187	246
29	220	109	106	204	140	73	138	158
30	244	151	184	158	11	247	135	192
31	236	227	213	174	139	169	8	73
32	1	1	1	1	1	1	1	1

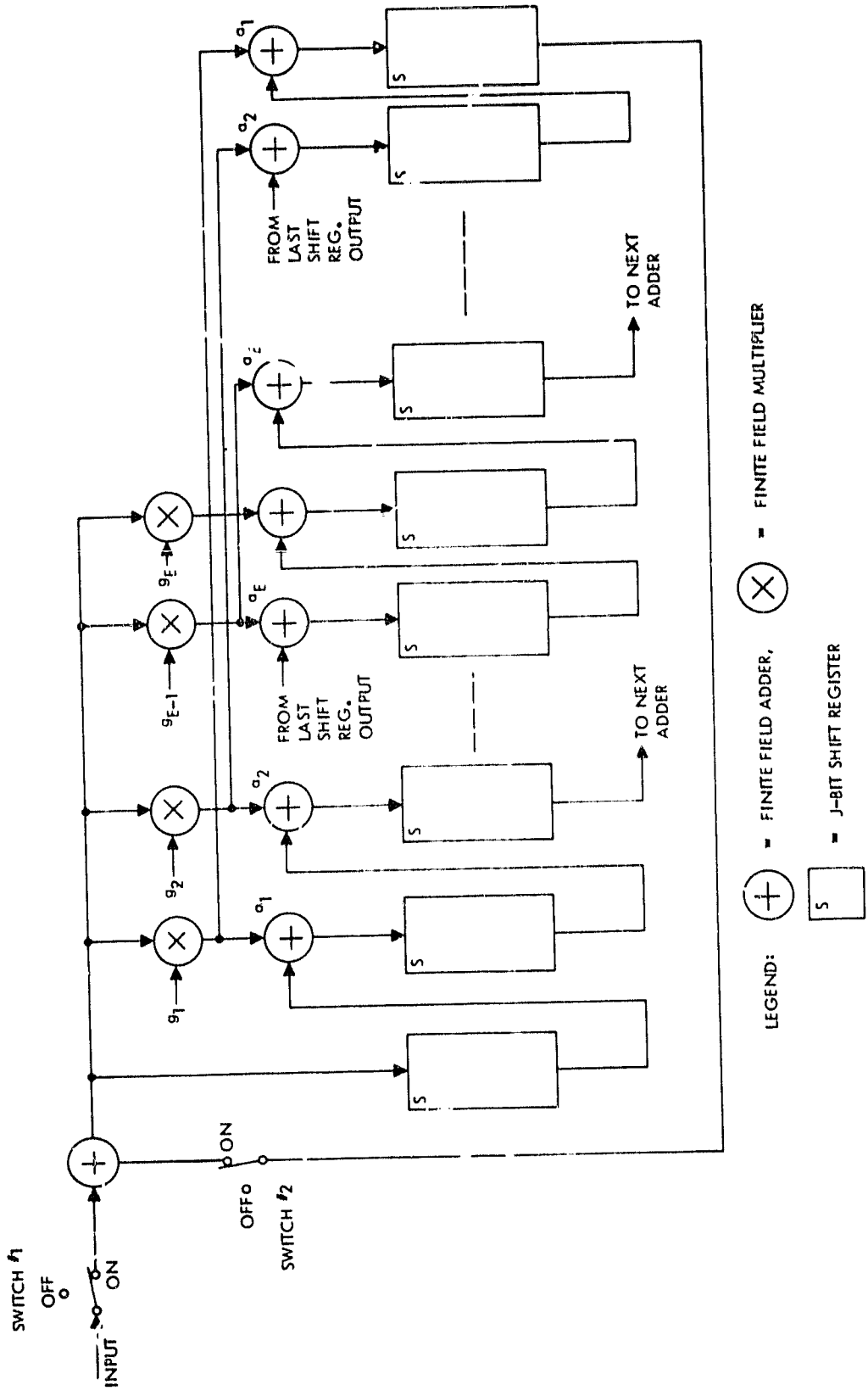
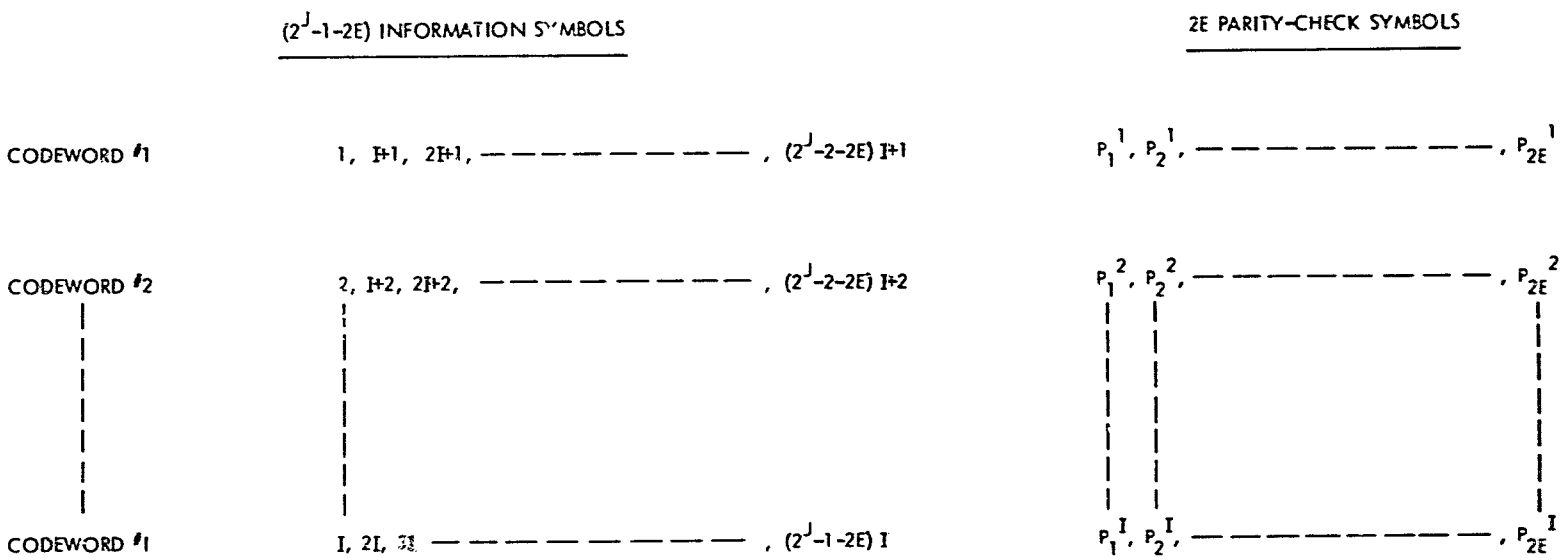


Figure 2. A Block Diagram of a $(2^J-1, 2^J-1-2E)$ RS Encoder Using a Generator Polynomial With Symmetrical Coefficients.

Legend: ⊕ = Finite Field Adder, ⊗ = Finite Field Multiplier,
[S] = J-bit Shift Register



ORDER OF SYMBOL TRANSMISSION

1, 2, — — — , I, I+1, — — — , 2I, 2I+1, — — — , (2^J-2-2E) I+1 , (2^J-2-2E) I+2,

— — — — —, (2^J-1-2E) I, P_1^1, P_1^2, — — —, P_1^I, P_2^1, — — —, P_2^I, — — ,

P_{2E}^1, P_{2E}^2, — — —, P_{2E}^I

Figure 3. Code Array Structure and Order of Symbol Transmission For Type B Interleaving, Where Interleaving Level = I

depth used. Thus I code words make up such an interleaved code block. After the information symbols are transmitted, the parity-check symbols of each interleaved code word are then transmitted.

If interleaving is used, then the encoder logic structure is the same as shown in Fig. 1, except now each J-bit shift register is replaced by an I x J-bit shift register. As an example, a block diagram of a (255,223) RS encoder with interleaving level I and generator polynomial

$$g(x) = \prod_{i=112}^{143} (x-\alpha^i)$$

where $\alpha = 2$ in $GF(2^8)$, which is generated by the primitive polynomial

$$x^8 + x^4 + x^3 + x^2 + 1,$$

is shown in Figure 4. Note that a generator polynomial with symmetrical coefficients is used here to save multipliers.

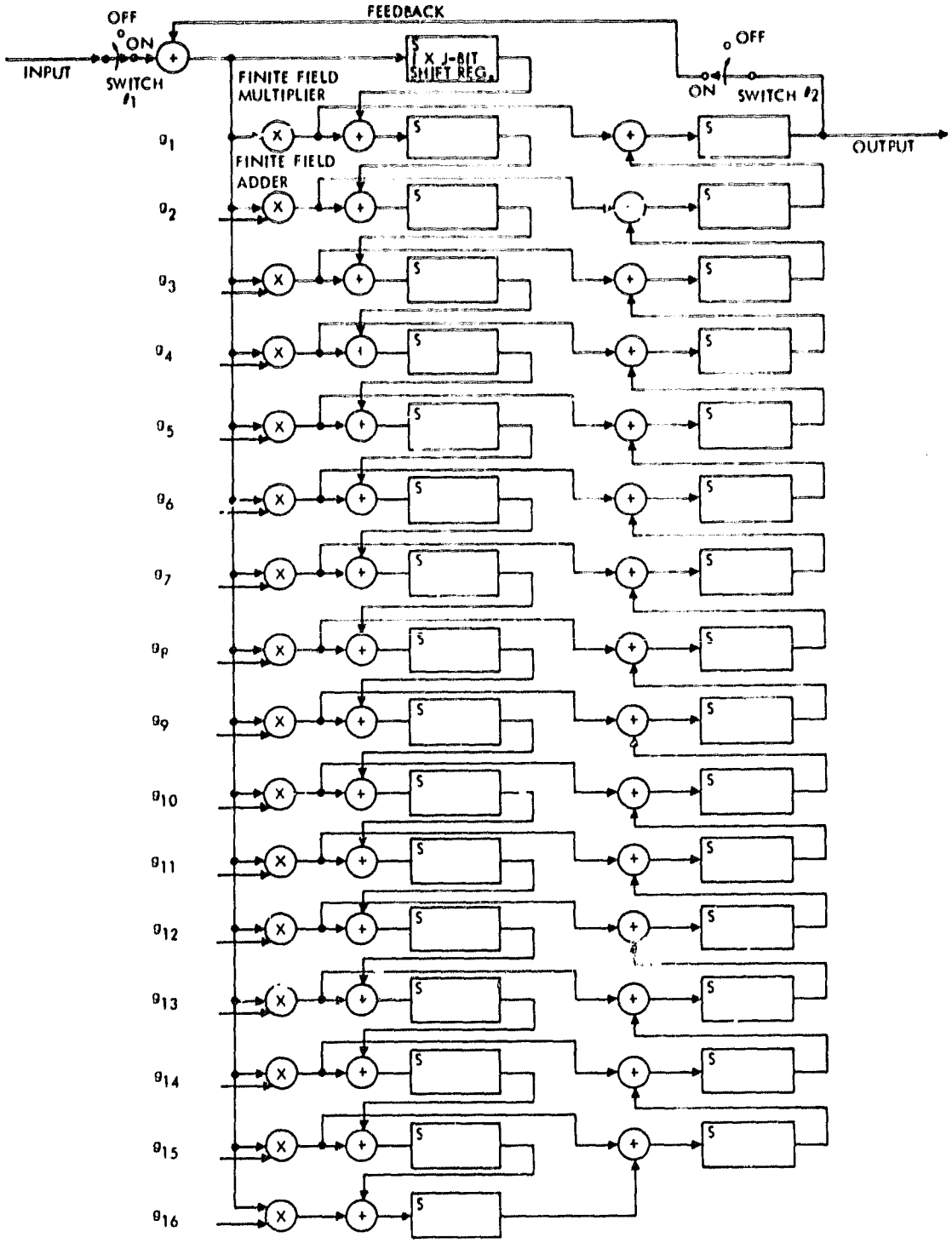


Figure 4. A Block Diagram of a (255,223) RS Encoder With Interleaving Level I and

$$g(x) = \prod_{i=112}^{143} (x-\alpha^i)$$

SECTION IV

SYMBOL-SLICE VLSI RS ENCODER ARCHITECTURE

A finite field multiplication is a quite complicated operation. There are basically three techniques for implementing a finite field multiplication. The first technique is to use log and antilog tables stored in read-only memories (ROM's) (Ref. 4). The second technique is to use a linear feedback shift register type of approach (Ref. 10). The third technique is to use the property of the trace in a finite field to form a smaller ROM look-up table (Ref. 11). Due to the advent of LSI ROM technology, techniques 1 and 3 are usually used in an RS encoder design optimized for discrete IC's. As an example a 400 KHZ (255,233) RS encoder using the Berlekamp's approach (Ref. 11) requires only around 40 CMOS IC's.

When one is interested in further drastic reduction of the power and size of an RS encoder for high speed applications, one has to consider VLSI implementations. An RS encoder design optimized for discrete IC's usually does not have a modular structure. Hence when one uses such an architecture for VLSI layout, one has the following problems:

(1) The design is too big to be put on one chip.

(2) If a multichip approach is used, then one needs several chip designs, where each chip has an impractical number of input/output pins.

(3) The design is not modular. Therefore the design is not easy to adapt to other RS code parameters.

Hence there is a need to find a VLSI logic structure which can alleviate the above problems.

The repetitive architecture of the RS encoders shown in Figs. 1, 2, and 4, suggested that a symbol-slice type of VLSI chips, each one consisting of a fixed portion of the encoder, may be cascaded to form a complete RS encoder. Also to reduce the VLSI chip size, RS encoders using generator polynomials with symmetrical coefficients are preferred. Hence we will put emphasis on this type of VLSI RS encoder.

As an example, we will design a VLSI encoder chip for a 255-symbol, 8-bit per symbol, 16-error-correcting, RS code with an interleaving level of 5. The primitive polynomial used is

$$x^8 + x^4 + x^3 + x^2 + 1$$

The generator polynomial for this RS code is

$$g(x) = \prod_{i=112}^{143} (x-\alpha^i)$$

where $\alpha = 2$. The coefficients of this g(x) are given in the first column of Table 3. The encoder logic structure for the above RS code parameter is identical to the one shown in Fig. 4, except now I = 5. There are several ways of partitioning the RS encoder into four sections. One way which requires a minimum of input/output pins is to include four rows of logic shown in Fig. 4 into one section. Each section is then realized by a universal VLSI RS encoder chip. Another way to partition the RS encoder shown in Fig. 4 into four sections is to include 8 rows of logic in each column into one section. Each section is then realized by

a universal VLSI RS encoder chip. These VLSI RS encoder configurations are described as follows.

4.1 VLSI RS ENCODER USING THE ROW PARTITIONING TECHNIQUE

The logic structure of the universal VLSI RS encoder chip using the row partitioning technique is shown in Fig. 5. The entire VLSI encoder system, which consists of four identical VLSI RS encoder chips cascaded and properly interconnected together is shown in Fig. 6. Each VLSI RS encoder chip has 24 pins. A detailed description of the VLSI RS encoder chip and the entire VLSI RS encoder system is described as follows:

4.1.1 Generator Polynomial Coefficients Table

Since a generator polynomial g(x) with symmetrical coefficients is used, the coefficients of x^0 is always 1. Hence there is no need for a multiplier to operate on this coefficient (see Fig. 4). Consequently if the new generator polynomial is used, then one only needs E/N multipliers on each VLSI chip, where E is the error correcting capability of the code and N is the total number of chips required in a VLSI encoder system. In the design example, E = 16 and N = 4. Hence 4 multipliers are used on each VLSI RS encoder chip. To make the VLSI chip universal, all distinct coefficients except 1 of the generator polynomial are stored in a read-only memory on the chip. In general, an ExJ-bit table is needed. In the design example E = 16, J = 8. Hence a 16 x 8-bit table is selected. The outputs of an ExJ-bit table is fed into N, E/N-to-one multiplexors. The outputs of the multiplexors are selected by $\log_2 N$ input pins called the "chip select" or "G select" pins. These outputs are then fed into the inputs of the E/N multipliers. In the design example two "G select" pins (pins 22 and 23) and four, four-to-one multiplexors are used. The 16 x 8 table and the multiplexors can easily be

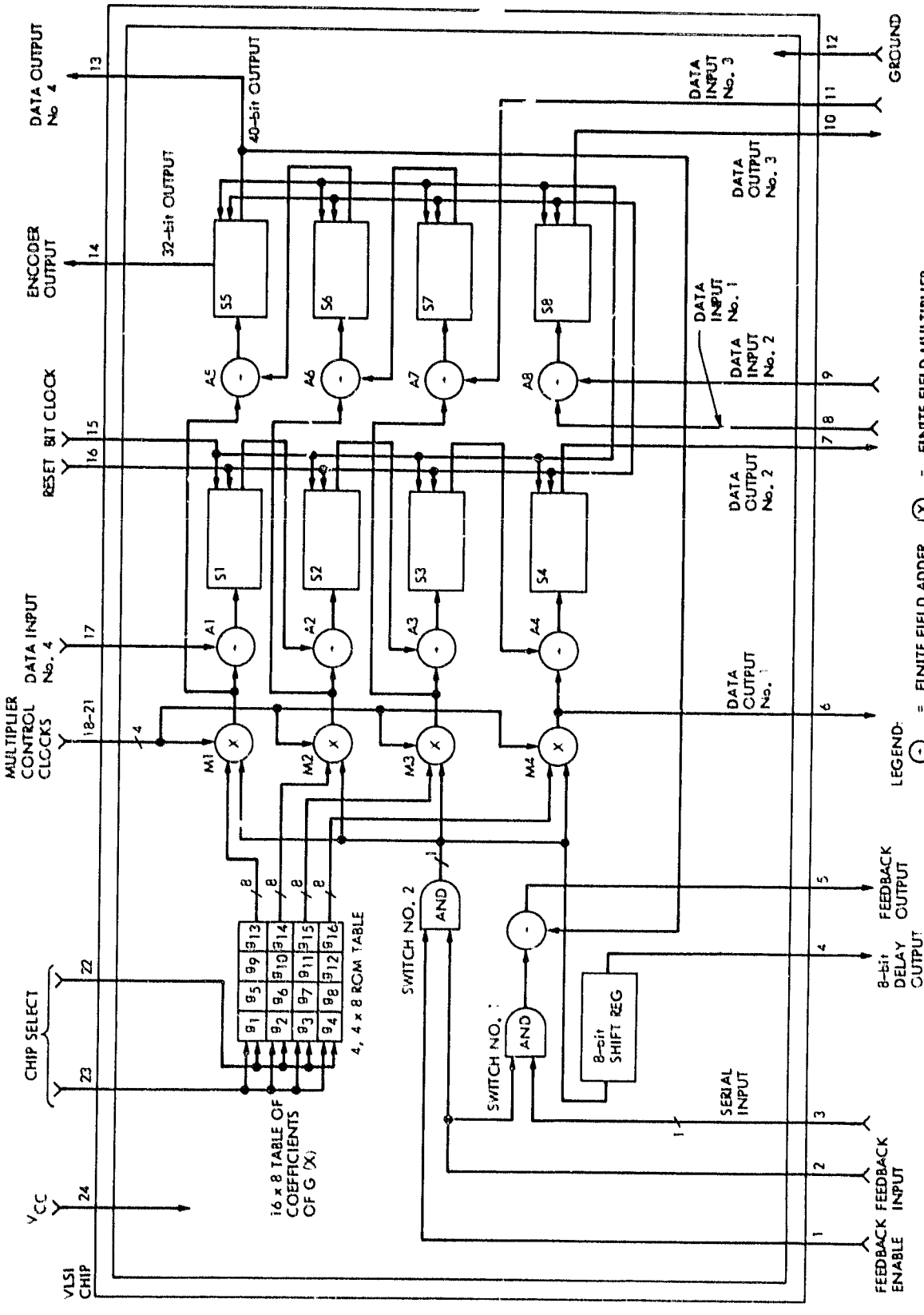


Figure 5. VLSI RS Encoder Chip Logic Structure (Row Partitioning).

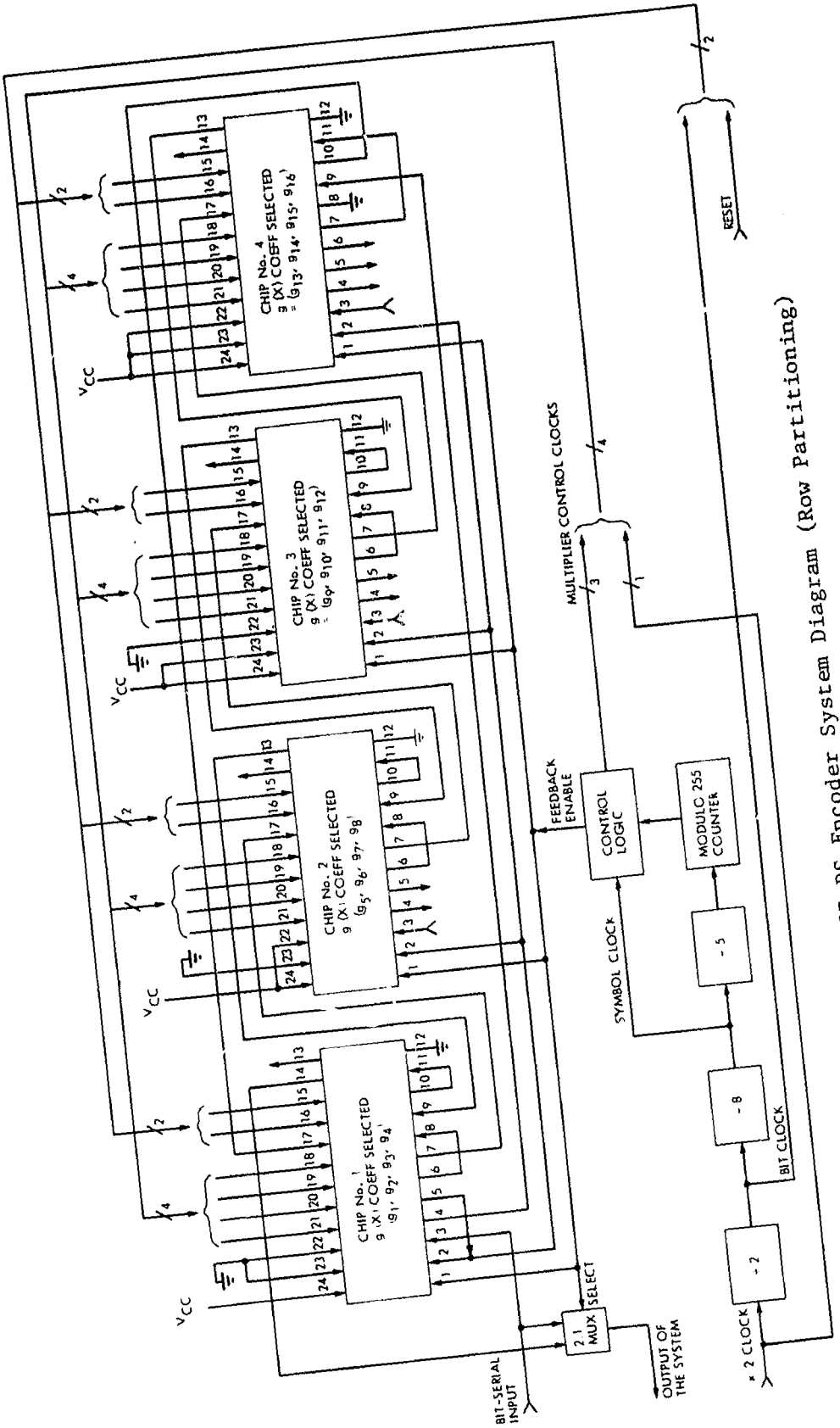


Figure 6. VLSI RS Encoder System Diagram (Row Partitioning)

implemented by four, 4 x 8 ROM, with G select signals as the address control lines of each ROM. The coefficients of g(x) selected on each chip are shown in Fig. 6.

4.1.2 Finite Field Multiplier

Next we will discuss the architecture of the finite field multiplier. To connect the multiplier properly between chips and at the same time minimize the number of input/output pins used, a linear feedback shift register type of multiplier (Ref. 9) rather than a ROM table look up type of multiplier (Ref. 4) is adopted. The multiplier used is of a serial-parallel type. The logic structure of the multiplier is shown in Fig. 7. The J-bit generator polynomial coefficient is read out from the ExJ-bit ROM table and fed into the multiplier J-bit in parallel whereas the other input, generated by the feedback input (pin 2) "ANDed" with the feedback enable (pin 1), is fed into the multiplier bit-by-bit in serial.

The output of the multiplier is loaded into an 8-bit shift register in parallel at the end of every 8th bit clock (1 symbol clock time). The parallel data is serialized by this shift register. The most significant bit (MSB) output of this shift register is added with the MSB of the 40-bit shift register, which is either on the same chip or on a different chip, and the resulting data is shifted into the least significant bit (LSB) of the next 40-bit shift register. The adder is implemented by a two-input EXCLUSIVE-OR gate (there are eight EXCLUSIVE-OR gates on each chip). Each adder takes an input from a multiplier output which is either on the same chip or on a different chip, depending on the coefficients of the generator polynomial.

4.1.3 Input and Feedback Control Switches

The switch #1 shown in Fig. 4 is implemented by an AND gate on the chip with the bit-serial data input (pin 3) and the feedback enable signal as the

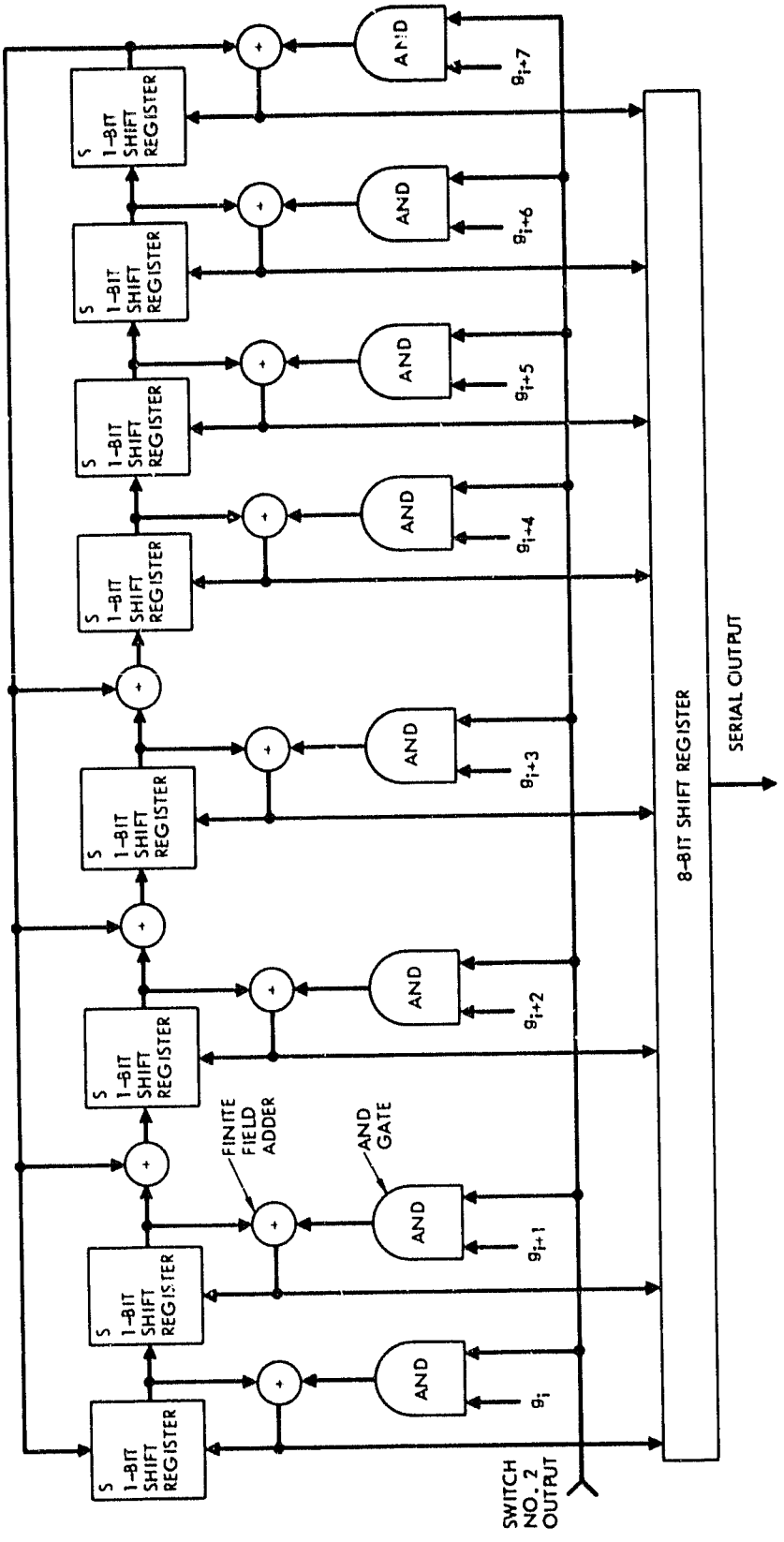


Figure 7. Logic Structure of a Serial-Parallel Finite Field Multiplier Which Performs $\alpha^i * \alpha^j$ modulo $\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$

two inputs. The feedback enable signal is provided by an external modulo 255 counter which is driven by a clock equal to the bit clock divided by 40 (see Fig. 6). This signal is true when the counter is counting from 1 to 223; otherwise it is false. The output of switch #1 is added with the MSB of the 40-bit shift register S5 output on the same chip to generate the feedback output signal (pin 5). This signal is redundant in all but the first chip (see Fig. 6).

The switch #2 shown in Fig. 4 is implemented by an AND gate on the chip with the feedback enable and feedback input signals as the two inputs. The feedback input signals on all chips (pin 2) are connected to the feedback output signal (pin 5) on the first VLSI chip. The output of switch #2 is fed into all multipliers on the chip bit-by-bit in serial.

4.1.4 Input/Output Data Connections

There are eight input/output lines on each chip. Of these eight lines, four lines (pins 8, 9, 11, 17) are input lines and the remaining are output lines (pins 6, 7, 10, 13). Pin 8 is normally connected to pin 6 on the same chip except for the last chip, where pin 8 is grounded. Pins 7 and 9 are normally connected to pins 17 and 13, respectively on the next chip except for the last chip, where pin 7 is connected to pin 11 on the same chip and pin 9 is connected to pin 4 on the first chip. Thus one has a railroad type of data connections between chips. The reason for connecting pin 4 on the first chip to pin 9 on the last chip is a consequence of an inherent 8-bit multiplier delay. To replace the multiplier on the x^0 position by the switch #1 output, one needs to delay this output also by 8 bits to line up the bits. For other chips besides the first chip, the 8-bit register outputs are not used.

The encoder output is taken 8 bits earlier from the MSB of the last 40-bit shift register on the first chip. This is because when the last bit of the last symbol in the information part of the code is shifted into the encoder, the contents of each multiplier are loaded into the 8-bit output shift registers waiting to be added with the MSB of the 40-bit shift registers. These 8-bit symbols in the multiplier output registers actually belong to the fifth code word in the interleaved code array. However the parity-check symbol of the first code word is already being computed and now sitting 8 bits from the MSBs of each 40-bit shift registers. Hence the 32-bit output (pin 14) of the last 40-bit shift register on the last chip is the output of the VLSI RS encoder system. This output is taken when the external modulo 255 counter is counting from 224 to 255. Since at these times switches #1 and #2 are turned off, the entire VLSI encoder system is behaving like an 1x2ExJ-bit (e.g., 5 x 32 x 8-bit in the design example) shift register. Thus the 5 x 32, 8-bit parity-check symbols are read out from the VLSI RS encoder bit-by-bit in serial and appended to the 5 x 223, 8-bit information symbols.

4.2 VLSI RS ENCODER USING THE COLUMN PARTITIONING TECHNIQUE

The logic structure of the universal VLSI RS encoder chip using the column partitioning technique is shown in Fig. 8. Note that this chip is very similar to the one shown in Fig. 5 except now one has to provide output pins to all multipliers and input pins to all adders on the chip. Hence this chip uses 6 more input/output pins than the one shown in Fig. 5. The entire VLSI RS encoder system using this logic structure is similar to the one shown in Fig. 6 except now the interchip connections between adders and multipliers are of the pyramid type shown in Fig. 9.

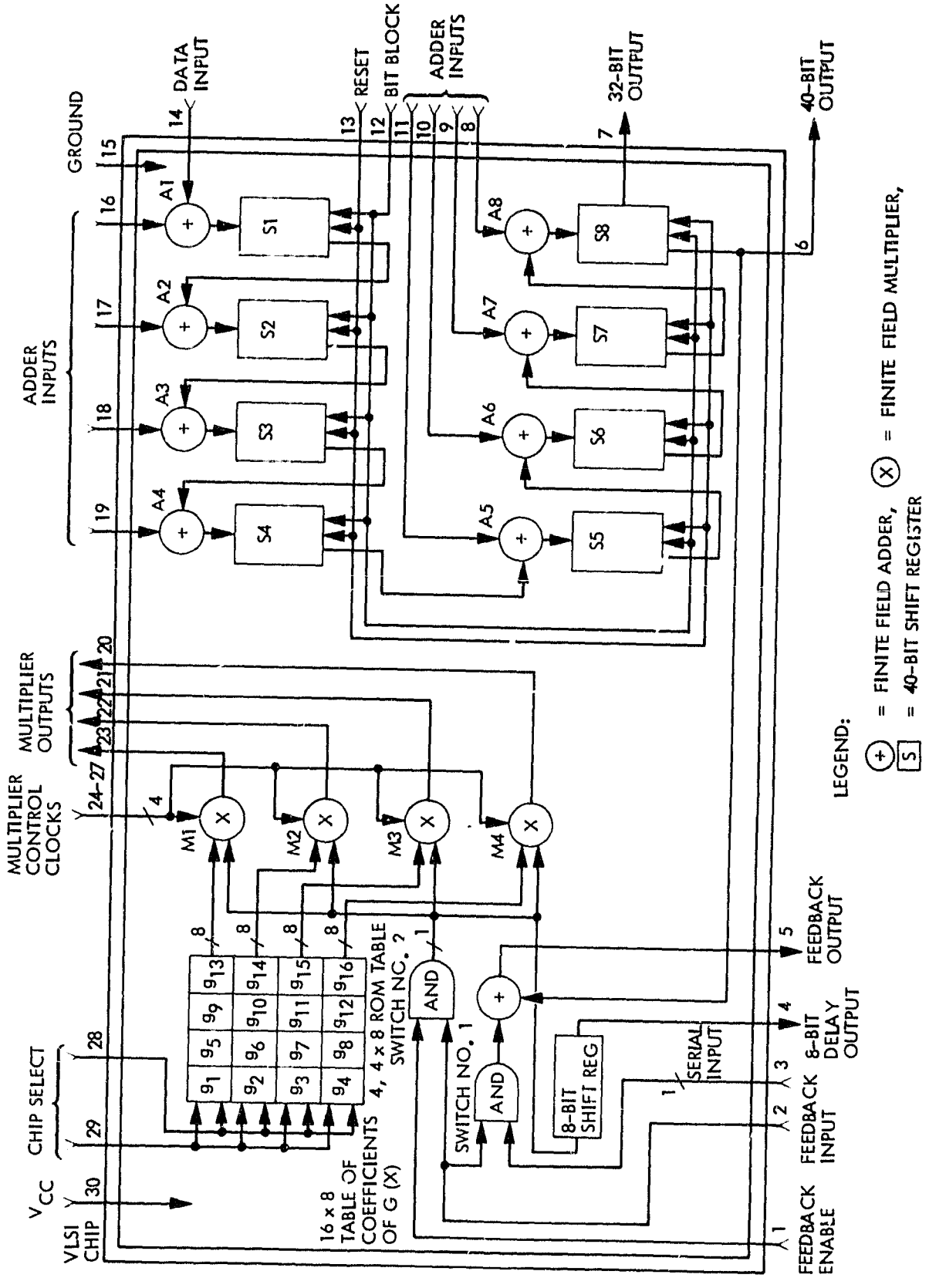


Figure 8. VLSI RS Encoder Chip Logic Structure (Column Partitioning).

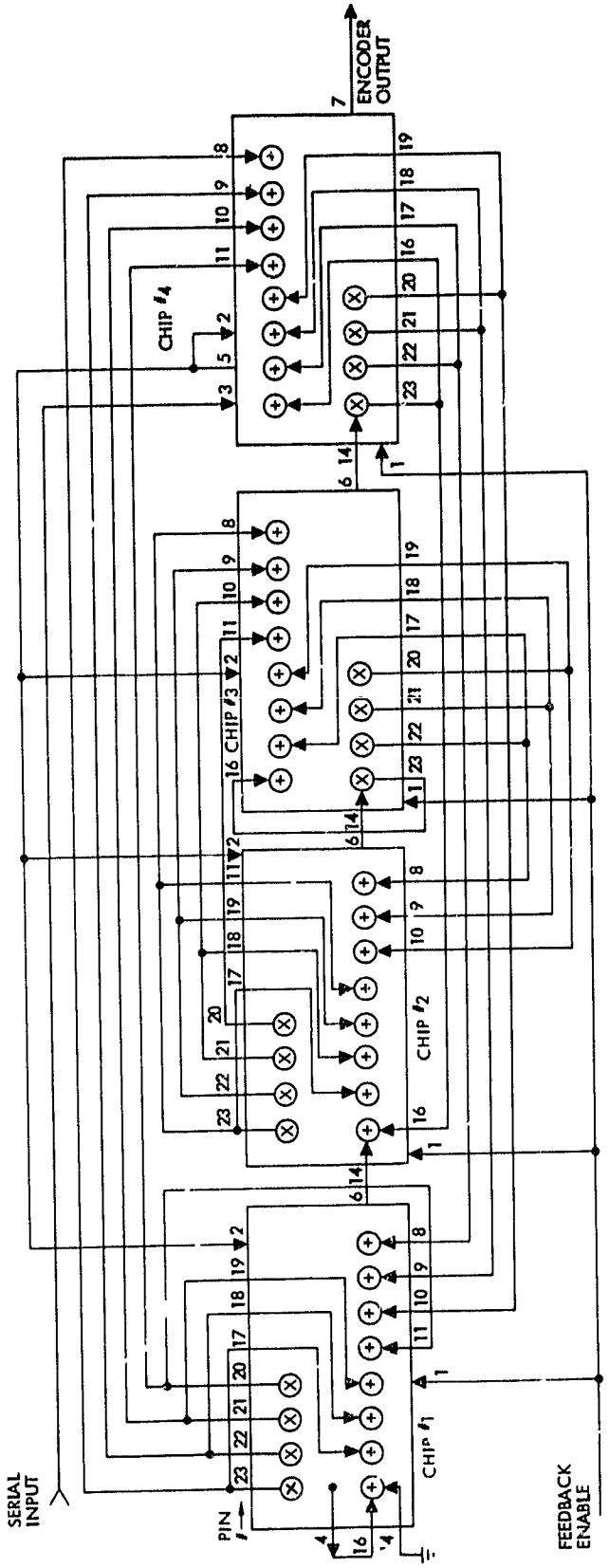


Figure 9. Inter-Chip Connection Diagram (Column Partitioning).

The 40-bit shift registers in the above two versions of the VLSI RS encoder chips can be replaced by random access memories (RAMs). In this case, a write-after-read operation should be performed during each bit time from the same location just read out from the RAM to simulate the shift register operation. Consequently, this version will not operate as fast as the shift register version. The first advantage of using the RAM approach is that interleaving level control can easily be incorporated into the RAM address control logic. Of course, input pins must be provided to select the interleaving level. Hence a more flexible VLSI RS encoder chip can be obtained using the RAM approach.

The second advantage of using the RAM approach is that a RAM cell usually occupies a smaller chip area than that of a static shift register. Hence a smaller VLSI RS encoder chip size can be obtained using the RAM approach. It is estimated that it is impossible to put the entire shift register version of the VLSI RS encoder chip design on a 235x235 mils CMOS/bulk VLSI chip using a 7 μm standard cell approach on all logic. However, if one uses custom RAM and ROM cells design to implement the 40-bit static shift registers and the 16x8 table and 7 μm standard cell design for the rest of the logic, then it is possible to have a one-chip design. Of course, the entire RS encoder chip design can easily be put on a smaller chip if a VLSI technology (say 3 μm standard cell approach) is used.

SECTION V

PERFORMANCE OF THE VLSI RS ENCODER SYSTEM

For design verification, both the shift register version and the RAM version of the VLSI RS encoder chip and VLSI RS encoder system are implemented using discrete CMOS IC's and are now operational. The throughputs of these two versions are 800K bits/sec for the shift register version and 200K bits/sec for the RAM version. These throughputs are expected to go much higher if the actual VLSI encoder chips are used.

One technique to improve the VLSI RS encoding speed is to multiplex the RS encoder chips. One type of multiplexing is to set the RS encoder chip interleaving level selection in the RAM approach to 1 (no interleaving) such that for a N-level of interleaving, a N-fold parallelism can be achieved. This scheme is illustrated in Fig. 10. Note that each row of the RS encoder chips in Fig. 10 are used to encode a RS code word corresponding to the one shown in each row of the code array structure in Fig. 3.

Another type of multiplexing is to use the relationship that if

$$f(x) = f_1(x) + f_2(x) + \cdots + f_N(x)$$

then

$$\frac{f(x)}{g(x)} = \frac{f_1(x)}{g(x)} + \frac{f_2(x)}{g(x)} + \cdots + \frac{f_N(x)}{g(x)}$$

can be realized by implementing $f_i(x)/g(x)$ for $i = 1, 2, \cdots, N$ in parallel and then summing the results from these parallel operations. Thus if one treats each

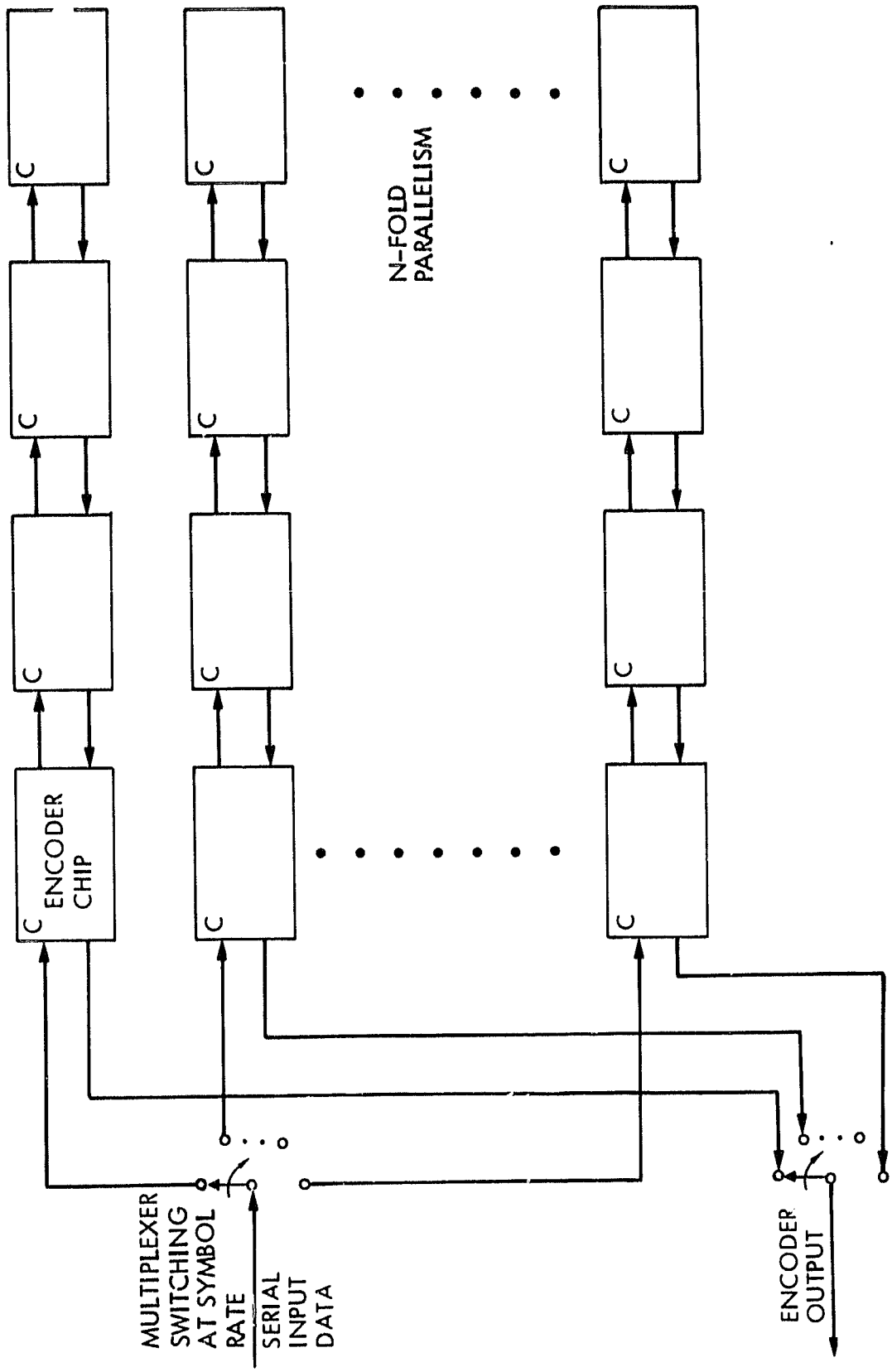


Figure 10. Throughput Improvement by Multiplexing VLSI RS Encoder Chips (Row Partitioning Version).

$f_i(x)$ as a polynomial, whose coefficients are selected from every N^{th} incoming symbols starting from the i^{th} symbol, where $i = 1, 2, ---, N$, then one can select the encoder chip interleaving level to 1 and use the logic structure similar to the one shown in Fig. 10 to realize the $f_i(x)/g(x)$ operation. The output of each row of encoder chips in this case needs to be properly delayed and summed to generate the encoder output $f(x)/g(x)$.

Another technique to improve the VLSI RS encoding speed is to process the J-bit incoming symbols in parallel. Parallel adders and multipliers are needed in this configuration. A throughput improvement of J times can be achieved using this approach. The disadvantages of this technique are:

(1) A lot of input/output pins are needed for the parallel data paths.

(2) A larger chip size is required to implement the VLSI RS encoder.

SECTION VI

CONCLUSIONS

We have just shown the logic structure of a symbolic-slice VLSI RS encoder chip and the VLSI RS encoder system built by these chips. A design example has been given for a (255,223) VLSI RS encoder chip and VLSI RS encoder system. It has been shown that an RS encoder consisting of four identical CMOS VLSI RS encoder chips connected together may replace around 40 CMOS IC's required by an encoder design optimized for discrete IC's. Besides the size advantage, the VLSI RS encoder also has the potential advantages of requiring less power and having a higher reliability. Finally, it is shown that such chips can easily be multiplexed to improve the encoding speed. The symbol-sliced logic structure presented in the report could also be applied to design VLSI RS decoders [12]. A separate report on this subject will be provided.

REFERENCES

[1] Reed, I.S., and Solomon, G., "Polynomial Codes Over Certain Finite Fields," J. Soc. Indust. Appl. Math., 8, pp. 300-304.

[2] Forney, G.D., Concatenated Codes, The MIT Press, Cambridge, Mass., 1966.

[3] Odenwalder, J.P., "Optimum Decoding of Convolutional Codes," Ph.D. dissertation, Syst. Sci. Dept., Univ. of Calif., Los Angeles, 1970.

[4] Odenwalder, J.P., et al. "Hybrid Coding System Study," submitted to NASA Ames Research Center by Linkabit Co., San Diego, Calif., Final Report, Contract No. NAS-2-6722, Sept. 1972.

[5] 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, CA., June 1974.

[6] Rice, R.F., "Potential End-to-End Imaging Information Rate Advantages of Various Alternative Communication Systems," JPL Publication 78-52, June 15, 1978.

[7] Hauptschein, A., "Practical, High Performance Concatenated Coded Spread Spectrum Channel for JTIDS," NTC '77, pp. 35:4-1 to 4-8.

[8] Liu, K.Y., and Lee, J., "An Experimental Study of the Concatenated Reed-Solomon/Viterbi Channel Coding System Performance and its Impact on Space Communications," JPL Pub. 81-58, Pasadena, CA, Aug. 15, 1981; also in Proc. Nat. Telecommun. Conf., 1981.

[9] Liu, K.Y., and Woo, K.T., "The Effects of Receiver Tracking Phase Error on the Performance of the Concatenated Reed-Solomon, Viterbi Channel Coding System," in Proc. Nat. Telecommun. Conf., 1980, pp. 51.5.1-51.5.5; also JPL Pub. 81-62, Pasadena, CA, September 1, 1981.

[10] Peterson, W.W., and Weldon, E.J., Jr., Error Correcting Codes, The MIT Press, 1972.

[11] Berlekamp, E.R., "Better Reed-Solomon Encoders," presented at Calif. Inst. of Tech. EE Seminar, Pasadena, CA., Dec. 12, 1979.

[12] Liu, K.Y., "Architectures for VLSI Design of Reed-Solomon Decoders," presented at IEEE 1st Int. Workshop on VLSI in Commun., Santa Barbara, CA, Sept. 1-4, 1981.

APPENDIX

VLSI REED-SOLOMON ENCODER FOR THE PROPOSED NASA/ESA TELEMETRY CHANNEL CODING STANDARD

After the completion of the VLSI RS encoder project, the following set of RS code parameters have been proposed in the NASA/ESA telemetry channel coding standard (Ref. A1):

(1) primitive polynomial

$$x^8 + x^7 + x^2 + x + 1.$$

(2) generator polynomial

$$g(x) = \prod_{i=112}^{143} \left[x - (\alpha^{11})^i\right]$$

where α satisfies the equation

$$x^8 + x^7 + x^2 + x + 1 = 0.$$

(3) Number of bits per symbol (J) = 8.

(4) Number of symbols per code word (N) = 255.

(5) Number of correctable symbol errors (E) = 16.

(6) Interleaving depth (I) = 5.

Note that the foregoing RS code parameters are those used in the design example in Section IV with two exceptions. These are the primitive polynomial (i.e., the field generator polynomial over GF(2) and the code generator polynomial. These polynomials were selected by Berlekamp in an architecture which minimizes the discrete ICs. Reference A2 provides a detailed description of such an architecture. To adapt to this new set of RS code parameters, one

needs to use a new field element table as well as a new table of generator polynomial coefficients. These are given in Tables A1 and A2 respectively.

Thus, to design a VLSI RS encoder for the new code parameters, one only needs to replace the generator polynomial coefficients ROM table shown in Figure 5 by the coefficients given in Table A2 under the heading α^{11}. Also one needs to replace the finite field multiplier shown in Figure 7 by the multiplier shown in Figure A1.

Table A1. Elements versus exponents in a finite field $GF(2^8)$ generated by the primitive polynomial $x^8 + x^7 + x^2 + x + 1$.

ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT	ELEMENT	EXPONENT
0	*	1	0	2	1	3	99	4	2	5	198
6	100	7	106	8	3	9	205	10	199	11	188
12	101	13	126	14	107	15	42	16	4	17	141
18	206	19	78	20	200	21	212	22	189	23	225
24	102	25	221	26	127	27	49	28	108	29	32
30	43	31	243	32	5	33	87	34	142	35	232
36	207	37	172	38	79	39	131	40	201	41	217
42	213	43	65	44	190	45	148	46	226	47	180
48	103	49	39	50	222	51	240	52	128	53	177
54	50	55	53	56	109	57	69	58	33	59	18
60	44	61	13	62	244	63	56	64	6	65	155
66	88	67	26	68	143	69	121	70	233	71	112
72	208	73	194	74	173	75	168	76	80	77	117
78	132	79	72	80	202	81	252	82	216	83	138
84	214	85	84	86	66	87	36	88	191	89	152
90	149	91	249	92	227	93	94	94	181	95	21
96	104	97	97	98	40	99	186	100	223	101	76
102	241	103	47	104	129	105	230	106	178	107	63
108	51	109	238	110	54	111	16	112	110	113	24
114	70	115	166	116	34	117	136	118	19	119	247
120	45	121	184	122	14	123	61	124	245	125	164
126	57	127	59	128	7	129	158	130	156	131	157
132	89	133	159	134	27	135	8	136	144	137	9
138	122	139	28	140	234	141	160	142	113	143	90
144	209	145	29	146	195	147	123	148	174	149	10
150	169	151	145	152	81	153	91	154	118	155	114
156	133	157	161	158	73	159	235	160	203	161	124
162	253	163	196	164	219	165	30	166	139	167	210
168	215	169	146	170	85	171	170	172	67	173	11
174	37	175	175	176	192	177	115	178	153	179	119
180	150	181	92	182	250	183	82	184	228	185	236
186	95	187	74	188	182	189	162	190	22	191	134
192	105	193	197	194	98	195	254	196	41	197	125
198	187	199	204	200	224	201	211	202	77	203	140
204	242	205	31	206	48	207	220	208	130	209	171
210	231	211	86	212	179	213	147	214	64	215	216
216	52	217	176	218	239	219	38	220	55	221	12
222	17	223	68	224	111	225	120	226	25	227	154
228	71	229	116	230	167	231	193	232	35	233	83
234	137	235	251	236	20	237	93	238	248	239	151
240	46	241	75	242	185	243	96	244	15	245	237
246	62	247	229	248	246	249	135	250	165	251	23
252	58	253	163	254	60	255	183				

Table A2. Generator Polynomial Coefficients for

$$g(x) = \prod_{i=112}^{143} (x - \alpha^{ij}), \text{ where } j=1,7,11,13,19,23,37 \text{ and } 43.$$

DEG(X)	α^{j}=2	α^{7}=173	α^{11}=173	α^{13}=61	α^{19}=222	α^{23}=251	α^{37}=174	α^{43}=30
0	1	1	1	1	1	1	1	1
1	236	227	213	174	169	139	8	73
2	244	151	184	158	247	11	135	192
3	220	109	106	204	73	140	138	158
4	133	47	132	153	177	63	187	246
5	238	104	36	89	80	43	65	160
6	137	26	144	44	53	223	130	94
7	201	184	194	91	59	97	123	159
8	7	213	9	154	129	236	198	171
9	141	4	59	158	106	41	44	5
10	11	133	67	188	103	88	102	240
11	226	174	81	234	18	156	117	129
12	34	238	56	191	47	102	77	124
13	252	97	158	72	184	145	138	188
14	209	157	204	226	64	98	214	242
15	22	232	74	59	83	31	106	173
16	78	6	135	78	168	220	125	80
17	22	232	74	59	83	31	106	173
18	209	157	204	226	64	98	214	242
19	252	97	158	72	184	145	138	188
20	34	238	56	191	47	102	77	124
21	226	174	81	234	18	156	117	129
22	11	133	67	188	103	88	102	240
23	141	4	59	158	106	41	44	5
24	7	213	9	154	129	236	198	171
25	201	184	194	91	59	97	123	159
26	137	26	144	44	53	223	130	94
27	238	104	36	89	80	43	65	160
28	133	47	132	153	177	63	187	246
29	220	109	106	204	73	140	138	158
30	244	151	184	158	247	11	135	192
31	236	227	213	174	169	139	8	73
32	1	1	1	1	1	1	1	1

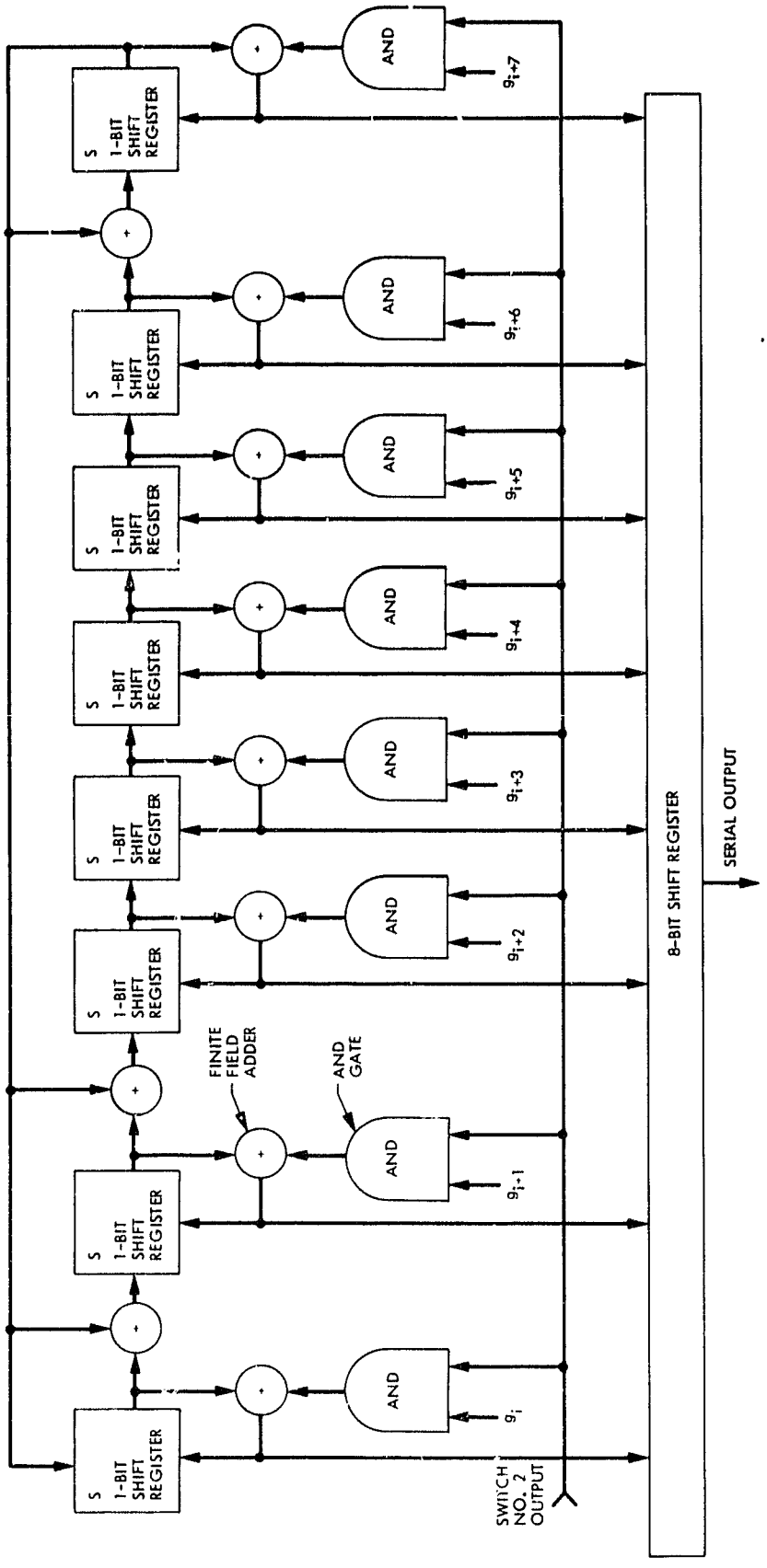


Figure A1. Logic structure of a serial-parallel finite field multiplier which performs $\alpha^i * \alpha^j$ modulo $\alpha^8 + \alpha^7 + \alpha^2 + \alpha + 1$.

REFERENCES

[A1] "Guidelines for Data Communications Standards: Space Telemetry Channel Coding (Issue-3)," prepared by the NASA/ESA Working Group for space data system standardization (NEWG), Jan. 1982.

[A2] Perlman, M. and Lee, J., "Reed-Solomon Encoders - Conventional Versus Berlekamp's Architecture," JPL IOM 3610-81-119 (internal document), July 10, 1981 (also to appear as JPL publication).