CCITT SG XV
Working Party XV / 1
Specialists Group on Coding for Visual Telephony

Document #464
March, 1989

Source : Japan
Title : Comparison between BCH code and Reed-Solomon code

1 Introduction

This document supports the proposal in Doc. #288 (Tokyo, Jan. 1988), Doc. #325 (Hague, Mar. 1988) and Doc. #412 (Florida, Dec. 1988) on BCH error correction technique for use in P×64 kbit/s visual telephony. The comparison between BCH code and Reed-Solomon code is described.

2 Comparison

The performance and feasibility of both codes are shown in Table 1. BCH code is superior in MTBF, decoding delay and multi-frame acquisition time, which are significant for practical use.

The complexity is evaluated based on computational steps, which is described in Annex 1.[1][2]

The capability of burst-error correction using BCH code is acquired based upon some study on cyclic codes.[1][3] 2-layer interleaved (511, 493) BCH code can correct 12 bits burst error without increasing multi-frame acquisition time. Meggitt decoder[1] is a scheme for burst error correction using BCH code. It can be implemented with linear feed back shift register circuits as well as a syndrome generator.

3 Conclusion

BCH code is desirable for use in visual telephony.

4 References

[1] Peterson & Weldon, " Error Correcting Codes ", MIT press, 1972.

[2] Polkinghorn, F., Jr. " Decoding of double and triple error correcting Bose-Chaudhuri Codes ", IEEE Trans., IT-12, 480-481, 1966.

[3] Tokiwa, K., Kasahara, M., Namekawa, T., " Burst-Error-Correction Capability of Cyclic Codes ", Trans. of IECE Japan, 66-A, 993-999, 1983.

Annex 1 Computational steps for decoding (including source program)
Annex 2 Burst error correction using Meggitt decoder

159

Item	(511, 493) BCH code	(1024, 992) RS code
block length	511 bits	1024 bits
num. of information bits	493 bits	992 bits
num. of check bits	18 bits	32 bits
efficiency	0.965	0.969
correctable error length	random 2 bits burst 6 bits (12 bits burst by 2-layer interleave)	random 2 bits burst 9 bits
MTBF for random error (at 2 Mbit/s)	24.1 sec (BER = 10^{-4}) 6.46 hours (BER = 10^{-5})	3.16 sec (BER = 10^{-4}) 49.2 min (BER = 10^{-5})
decoding delay (at 64kbit/s)	7.98msec (random error) 16.0msec (random & burst)	16.0msec (random & burst)
acquisition time (multi-frame of 8 blocks) (1 step acquisition) (at 64kbit/s)	64 msec	128 msec
complexity* of random error correction (with LUT)	1	5
check bits generator (hardware)	8 EX-OR 18 D-FF	4 32 EX-OR 4 32 D-FF 4 ROM (Total 2k byte)
syndrome generator (hardware)	6 EX-OR 18 D-FF	4 32 EX-OR 4 32 D-FF 4 ROM
specific hardware for burst error correction	52 EX-OR 18 D-FF a few gates	–

*complexity was evaluated based on computational steps. (See Annex 1)

Table 1 Comparison between (511,493) BCH and (1024, 992) RS

160

Annex 1 to Document # 464

Annex 1 to Document #464
March, 1989

Source : Japan
Title : Computational steps for decoding

1 Introduction

This annex shows the computational steps for the random error correction after the syndromes have been generated. It is supposed that the syndrome generator is implemented in dedicated hardware as supposed in Doc. #335. The syndrome generator can be implemented with the combination of some registers and some EXOR-gates for BCH code and Reed-Solomon code respectively. Furthermore, Look-up table is used to obtain the roots of the error locator equation.

The list of source program for BCH decoding is attached.

2 Procedure of random error correction

The errors are found by the following steps.

[BCH code]

The error locator equation can be written as follows.

$$X^2 + S_1X + (S_1^3 + S_3)/S_1 = 0 \qquad \text{Ⓐ}$$

where S_1 and S_3 denote the syndromes.

The above equation is expressed as

$$Y^2 + Y + (1 + S_3/S_1^3) = 0 \qquad \text{Ⓑ}$$

where $X = S_1Y$.

The roots of this euation can be derived from the Look-up table using the value of S_3/S_1^3 as the address of the table.

[Reed-Solomon code]

When the generation polynomial of Reed-Solomon code is defined as

$$G(X) = (X+\alpha)(X+\alpha^2)(X+\alpha^3)(X+\alpha^4),$$

the error locator equation can be written as follows.

$$X^2 + [(S_1S_4 + S_2S_3)/(S_1S_3 + S_2^2)]X + [(S_2S_4 + S_3^2)/(S_1S_3 + S_2^2)] = 0 \qquad \text{Ⓒ}$$

The above equation is expressed as

$$Y^2 + Y + [(S_1S_3 + S_2^2)(S_2S_4 + S_3^2)/(S_1S_4 + S_2S_3)^2] = 0 \qquad \text{Ⓓ}$$

where $X = [((S_1S_4 + S_2S_3)/(S_1S_3 + S_2^2)]Y$.

161

The roots of this euation can be derived from the Look-up table using the value of $[(S_1S_3 + S_2^2)(S_2S_4 + S_3^2) / (S_1S_4 + S_2S_3)^2]$ as the address of the table.

Next, the error values have to be calculated as :

$$e_1 = [(S_1\alpha^i + S_2)(S_1S_3 + S_2^2) / \{\alpha^j (S_1S_4 + S_2S_3)\}] \qquad ⑤$$

$$e_2 = [(S_1\alpha^j + S_2)(S_1S_3 + S_2^2) / \{\alpha^i (S_1S_4 + S_2S_3)\}]. \qquad ⑥$$

where α^j and α^i denote the locations of the errors.

3 Computational steps

Let wx be a x-th register and (wx) be a value stored in wx.

[BCH code]

The value of S_3 / S_1^3 is to be calculated.

Given : the syndromes $S_1 = \alpha^p$, $S_3 = \alpha^q$

1	p → w0		:	convert S_1 to its logarithm representation p using LUT1 (p is for $S_1 = \alpha^p$)
2	q → w1		:	convert S_3 to its logarithm representation q using LUT2 (q is for $S_3 = \alpha^q$)
3	(w0) + (w0) → w2	(mod 511)	:	calculate 2p (S_1^2)
4	(w2) + (w0) → w2	(mod 511)	:	calculate 3p (S_1^3)
5	− (w2) → w2	(mod 511)	:	calculate −3p ($1 / S_1^3$)
6	q + (w0) → w2	(mod 511)	:	calculate q − 3p (S_3 / S_1^3)
7	k → w3		:	look-up the relative error location (1) using LUT3 (k is for α^k) (the root of Ⓑ (1))
8	l → w4		:	look-up the relative error location (2) using LUT4 (l is for α^l) (the root of Ⓑ (2))
9	(w0) + (w3) → w3	(mod 511)	:	the error location (1) (the root of Ⓐ (1))
10	(w0) + (w4) → w4	(mod 511)	:	the error location (2) (the root of Ⓐ (2))

total steps 10 steps

[Reed-Solomon code]

The value of $[(S_1S_3 + S_2^2)(S_2S_4 + S_3^2) / (S_1S_4 + S_2S_3)^2]$ is to be calculated in order to find the error location. Next, the error values e1 and e2 are also to be calculated.

$$e_1 = [(S_1\alpha^i + S_2)(S_1S_3 + S_2^2) / \{\alpha^j (S_1S_4 + S_2S_3)\}]$$

$$e_2 = [(S_1\alpha^j + S_2)(S_1S_3 + S_2^2) / \{\alpha^i (S_1S_4 + S_2S_3)\}]$$

$\oplus$ denotes addition (EX-OR) in polynomial representation.

Given : the syndromes $S_1 = \alpha^p$, $S_2 = \alpha^q$, $S_3 = \alpha^r$, $S_4 = \alpha^s$

1	$p \rightarrow w0$		: convert S_1 to its logarithm representation p using LUT1 (p is for $S1 = \alpha^p$)
2	$q \rightarrow w1$		: convert S_2 to its logarithm representation q using LUT1 (q is for $S_2 = \alpha^q$)
3	$r \rightarrow w2$		: convert S_3 to its logarithm representation r using LUT1 (q is for $S_3 = \alpha^r$)
4	$s \rightarrow w3$		: convert S_4 to its logarithm representation s using LUT1 (q is for $S_4 = \alpha^s$)
5	$(w0) + (w2) \rightarrow w4$	(mod 255)	: calculate $p + r$ (S_1S_3)
6	$(w1) + (w1) \rightarrow w5$	(mod 255)	: calculate $q + q$ (S_2^2)
7	$(w1) + (w3) \rightarrow w6$	(mod 255)	: calculate $q + s$ (S_2S_4)
8	$(w2) + (w2) \rightarrow w7$	(mod 255)	: calculate $r + r$ (S_3^2)
9	$(w0) + (w3) \rightarrow w8$	(mod 255)	: calculate $p + s$ (S_1S_4)
10	$(w1) + (w2) \rightarrow w9$	(mod 255)	: calculate $q + r$ (S_2S_3)
11	$S_1S_3 \rightarrow w2$		: convert $p + r$ to its polynomial rep. S_1S_3 using LUT2 $(S_1S_3 = \alpha^{(p+r)})$
12	$S_2^2 \rightarrow w3$		: convert $q + q$ to its polynomial rep. S_2^2 using LUT2 $(S_2^2 = \alpha^{2q})$
13	$(w2) \oplus (w3) \rightarrow w4$		: calculate $S_1S_3 + S_2^2$
14	$f \rightarrow w4$		: convert $S_1S_3 + S_2^2$ to its logarithm rep. f using LUT1 (f is for $S_1S_3 + S_2^2 = \alpha^f$)
15	$S_2S_4 \rightarrow w2$		: convert $q + s$ to its polynomial rep. S_2S_4 using LUT2 $(S_2S_4 = \alpha^{(q+s)})$
16	$S_3^2 \rightarrow w3$		: convert $r + r$ to its polynomial rep. S_3^2 using LUT2 $(S_3^2 = \alpha^{2r})$
17	$(w2) \oplus (w3) \rightarrow w5$		: calculate $S_2S_4 + S_3^2$
18	$g \rightarrow w5$		: convert $S_2S_4 + S_3^2$ to its logarithm rep. g using LUT1 (g is for $S_2S_4 + S_3^2 = \alpha^g$)
19	$S_1S_4 \rightarrow w2$		: convert $p + s$ to its polynomial rep. S_1S_4 using LUT2 $(S_1S_4 = \alpha^{(p+s)})$

20	$S_2S_3 \rightarrow w3$		: convert q + r to its polynomial rep. S_2S_3 using LUT2 $(S_2S_3 = \alpha^{(q+r)})$
21	$(w2) \oplus (w3) \rightarrow w6$		: calculate $S_1S_4 + S_2S_3$
22	$h \rightarrow w6$		: convert $S_1S_4 + S_2S_3$ to its logarithm rep. h using LUT1 (h is for $S_1S_4 + S_2S_3 = \alpha^h$)
23	$(w6) + (w6) \rightarrow w7$	(mod 255)	: calculate h + h ($(S_1S_4 + S_2S_3)^2$)
24	$-(w7) \rightarrow w7$	(mod 255)	: calculate −2h ($1/(S_1S_4 + S_2S_3)^2$)
25	$(w4) + (w5) \rightarrow w2$	(mod 255)	: calculate f + g for $(S_1S_3 + S_2^2)(S_2S_4 + S_3^2)$
26	$(w2) + (w7) \rightarrow w2$	(mod 255)	: calculate f + g − 2h for $[(S_1S_3 + S_2^2)(S_2S_4 + S_3^2) / (S_1S_4 + S_2S_3)^2]$
27	$\alpha^l \rightarrow w2$		: look-up the error location in polynomial representation (1) using LUT3
28	$(w2) \oplus 1 \rightarrow w3$		: calculate the error location in polynomial representation (2) $\because \alpha^l + \alpha^m = 1$ where α^l and α^m are the roots of Ⓓ
29	$l \rightarrow w8$		: convert α^l to its logarithm rep. l using LUT1 $(\alpha^l \rightarrow l)$
30	$m \rightarrow w9$		: convert α^m to its logarithm rep. m using LUT1 $(\alpha^m \rightarrow m)$
31	$-(w4) \rightarrow w2$	(mod 255)	: calculate −f $(1/(S_1S_3 + S_2^2))$
32	$(w2) + (w6) \rightarrow w4$	(mod 255)	: calculate h − f $((S_1S_4 + S_2S_3) / (S_1S_3 + S_2^2))$
33	$(w4) + (w8) \rightarrow w2$	(mod 255)	: calculate h − f + l : the root of © (1) $\because \alpha^l (S_1S_4 + S_2S_3) / (S_1S_3 + S_2^2) = \alpha^i$
34	$(w4) + (w9) \rightarrow w3$	(mod 255)	: calculate h − f + m : the root of © (2) $\because \alpha^m (S_1S_4 + S_2S_3) / (S_1S_3 + S_2^2) = \alpha^j$
35	$(w0) + (w2) \rightarrow w5$	(mod 255)	: calculate p + i $(S_1\alpha^i)$
36	$S_1\alpha^i \rightarrow w5$		: convert p + i to its polynomial rep. $S_1\alpha^i$ using LUT2 $(S_1\alpha^i = \alpha^{(p+i)})$
37	$(w0) + (w3) \rightarrow w6$	(mod 255)	: calculate p + j $(S_1\alpha^j)$
38	$S_1\alpha^j \rightarrow w6$		: convert p + j to its polynomial rep. $S_1\alpha j$ using LUT2 $(S_1\alpha j = \alpha^{(p+j)})$
39	$S_2 \rightarrow w7$		: convert q to its polynomial rep. S_2 using LUT2 $(S_2 = \alpha^q)$
40	$(w5) \oplus (w7) \rightarrow w5$		: calculate $(S_1\alpha^i + S_2)$
41	$(w6) \oplus (w7) \rightarrow w6$		: calculate $(S_1\alpha^j + S_2)$
42	$u \rightarrow w5$		: convert $S_1\alpha^i + S_2$ to its logarithm rep. u using LUT1 $(S_1\alpha^i + S_2 = \alpha^u)$
43	$v \rightarrow w6$		: convert $S_1\alpha^j + S_2$ to its logarithm rep. v using LUT1 $(S_1\alpha^j + S_2 = \alpha^v)$

44	$-(w4) \rightarrow w4$	(mod 255)	: calculate $-(h-f)$ $((S_1S_3+S_2^2)/(S_1S_4+S_2S_3))$
45	$-(w2) \rightarrow w7$	(mod 255)	: calculate $-i$ $(1/\alpha^i)$
46	$-(w3) \rightarrow w8$	(mod 255)	: calculate $-j$ $(1/\alpha^j)$
47	$(w4)+(w7) \rightarrow w7$	(mod 255)	: calculate $-(h-f)-i$ $((S_1S_3+S_2^2)/\{\alpha^i(S_1S_4+S_2S_3)\})$
48	$(w4)+(w8) \rightarrow w8$	(mod 255)	: calculate $-(h-f)-j$ $((S_1S_3+S_2^2)/\{\alpha^j(S_1S_4+S_2S_3)\})$
49	$(w5)+(w8) \rightarrow w5$	(mod 255)	: calculate $u-(h-f)-j$ $(S_1\alpha^i+S_2)(S_1S_3+S_2^2)/\{\alpha^j(S_1S_4+S_2S_3)\}$ the error value (1) (logarithm) Ⓔ
50	$e_1 \rightarrow w7$		: convert $u-(h-f)-j$ to its polynominal representation e_1 using LUT2 the error value (1) (polynomial) Ⓔ
51	$(w6)+(w7) \rightarrow w6$	(mod 255)	: calculate $v-(h-f)-i$ $(S_1\alpha^j+S_2)(S_1S_3+S_2^2)/\{\alpha^i(S_1S_4+S_2S_3)\}$ the error value (2) (logarithm) Ⓕ
52	$e_2 \rightarrow w8$		: convert $v-(h-f)-i$ to its polynomial representation e_2 using LUT2 the error value (2) (polynomial) Ⓕ
	total steps		52 steps

4 Number of steps

BCH code	10 steps
Reed-Solomon code	52 steps

5 Conclusion

The computational steps for random error correction was shown.

Source program for BCH decoding

```
C-----------------------------------------------------------------------
C
C    BCH decoder for (511,493)
C    double error correction
C
C      SYND1 (IN)   : syndrome S1 ( polynomial )
C      SYND3 (IN)   : syndrome S3 ( polynomial )
C      IE1   (OUT)  : error location 1
C      IE2   (OUT)  : error location 2
C
C    Tables
C      EXPS1        : polynomial -> logarithm for S1
C      EXPS3        : polynomial -> logarithm for S3
C      SXI          : Look-up table for error location 1
C      SXJ          : Look-up table for error location 2
C
C-----------------------------------------------------------------------
      SUBROUTINE BCHDC(SYND1,SYND3,IE1,IE2)
      IMPLICIT INTEGER(A-Z)
C
      INTEGER EXPS1(0:511),EXPS3(0:511)
      INTEGER SXI(0:511),SXJ(0:511)
C
      COMMON / TAB / EXPS1,EXPS3,SXI,SXJ
C
C           SYNDROME SYND1 & SYND3 GIVEN !
C
            S1=EXPS1(SYND1)                              ! convert to log.
            S3=EXPS3(SYND3)                              ! convert to log.
C
            IF ((S1.NE.511).OR.(S3.NE.511)) THEN         ! error
              CALL DIV(S1,S3,SS)
              CALL ERLC(S1,SS,IE1,IE2)
            ELSE                                         ! no error
              IE1=-1
              IE2=-1
            ENDIF
C
      RETURN
      END
C-----------------------------------------------------------------------
C     S3/S1**3
C-----------------------------------------------------------------------
      SUBROUTINE DIV(S1,S3,SS)
      IMPLICIT INTEGER(A-Z)
C
      IF (S3.EQ.511) THEN                    ! uncorrectable error
        SS=511
      ELSE                                   ! correctable error
        SS1=MOD(S1*3,511)
        SS=S3-SS1
        IF (SS.LT.0) SS=SS+511
      ENDIF
C
      RETURN
      END
C-----------------------------------------------------------------------
C       ERROR LOCATION
C-----------------------------------------------------------------------
      SUBROUTINE ERLC(S1,SS,IE1,IE2)
      IMPLICIT INTEGER(A-Z)
      INTEGER EXPS1(0:511)EXPS3(0:511)
      INTEGER SXI(0:511),SXJ(0:511)
C
      COMMON / TAB / EXPS1,EXPS3,SXI,SXJ
C
      SRI=SXI(SS)
      SRJ=SXJ(SS)
      IE1=MOD(SRI+S1,511)
      IE2=MOD(SRJ+S1,511)
C
      RETURN
      END
```

Annex 2 to Document #464
March, 1989

Source : Japan
Title : Burst error correction using Meggitt decoder

A burst error correction method for BCH code can be implemented as follows.