A Realization of RS Code Encoding and Decoding in Software

Chenglin Miao [a], Tong Li [b], Xiaoling Wang , Huiming Wu

Department of Information Engineering, Academy of Armored Forces Engineering, AAFE Beijing, China

[a]13601187076@163.com,[b]644392162@qq.com

Keywords: RS code, Encoding, Decoding.

Abstract. The RS code is a kind of nor binary BCH codes, and it has strong capability of error correcting, which can correct not only random errors but also burst errors. Therefore, RS code may be widely used in modern communications systems. With the rapid development of software defined radio technology, encoding and decoding of RS code are implemented in common hardware platform. Generally speaking, the simulation experiment is based on FPGA (field programmable gate array) using VHDL-language or Verilog-language; in other ways it is based on DSP (digital signal processor) using C-language or assembly language. This paper will introduce the basic computation rules of RS codes firstly, and next will explain encoding and decoding process and its MATLAB simulation experiment. At last, we design a system of RS code and this paper will demonstrate the result of system in MATLAB software.

Introduction

The RS code is a kind of linear block codes, and it owns clearly coding efficiency and strong capability of error correcting. It can correct (*n*-*k*)/2 random error and (*n*-*k*)/2 burst errors. Compared to cyclic code, BCH code and other linear block codes, RS code has the strongest capability of error correcting. Although some semiconductor companies have produced dedicated chips for RS code encoding and decoding, these chips are too expensive. With accomplishing encoding and decoding in software method, on the basis of assuring calculating accuracy, the method can reduce cost efficiently.

Important Properties of RS code

In theory of RS code, the concept of domain is very important, which is called Galois field or finite field and is shorted as GF. As to $GF(2^4)$, there are 2*m* elements which are shown as $0,\alpha^0,\alpha^1\cdots\alpha^{13},\alpha^{14}$, where α is the root of primitive polynomial and the last sign α^{q-1} is 1. We operate all computation rules about RS codes in Galois field including GF plus, multiplication, division and comparing operation. For example, when *m*=4, its primitive polynomial is $p(x)=x^4+x+1$. Because α is the root of p(x), so $\alpha^4+\alpha+1=0$ that is $\alpha^4=\alpha+1$ in other words. As shown in Table 1, every sign in $GF(2^4)$,which is $0,\alpha^0,\alpha^1\cdots\alpha^{13},\alpha^{14}$, express 0,1,2,4,8,3,6,12,11,5,10,7,14,15,13,9,1 in decimal value.

The primitive polynomial is important for us to obtain all elements in Galois field. As to how to know primitive polynomial, it is pure mathematical problem. Now, corresponding relation of the order of Galois field and primitive polynomial can be shown in Table [1].

Table 1 some relation of Galois field

Element	Evaluation	Binary value	Decimal value
α^0	1	0001	1
α^1	α^1	0010	2
α^2	α^2	0100	4
…	…	…	…
α^{13}	$\alpha^3+\alpha^2+1$	1101	13

α^{14}	α^3+1	1001	9

The principle of RS code encoding

Before we study the circuit of encoding, the multiplier and divider should be introduced. The encoding circuit's basic structure is based on multiplier and divider.

Multiplier and divider. We define two polynomials, where

$A(x) = a_k x^k + a_{k-1}x^{k-1} \cdots + a_1 x + a_0$

$B(x) = b_k x^k + b_{k-1}x^{k-1} \cdots + b_1 x + b_0$

The following flow chart is about $A(x)B(x)$. The process of a multiplier is shown in Fig. 1.

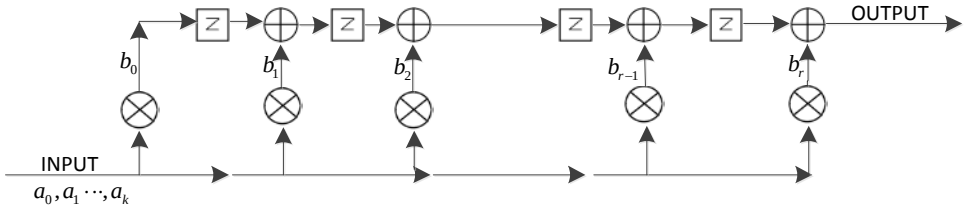

Fig. 1 The process of a multiplier

Step 1: We should initialize registers.

Step 2: When we firstly input a_k, which is the coefficient of the highest order of $A(x)$, into the multiplier system. And registers are assigned values of $a_k b_0, a_k b_1 \cdots, a_k b_r$, and the multiplier system's output is a_k.

Step 3: The second coefficient a_{k-1} is transmitted into the system, and the value of the m register adds with $a_{k-1}b_m$, while the result is sent to the next register. The value of the m register is updated by the last one. The multiplier system's output is $a_{k-1}b_r + a_k b_{r-1}$.

Step 4: According to the steps above, the processes are repeated until the multiplier system output constant term after right shifting $k+r+1$ character.

The following flow chart is about $A(x) / B(x)$. The process of a divider is shown in Fig. 2.

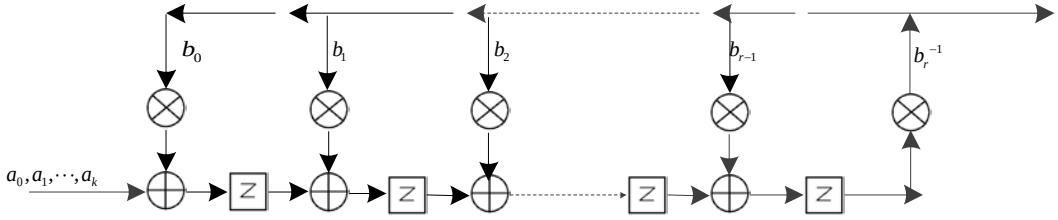

Fig. 2 The process of a divider

Step 1: At the beginning of the operation, we should reset all registers. The coefficient of the highest order of $A(x)$ is firstly sent into the most left register in divider system. After shifting r characters to right, 1 to 2t registers are assigned respectively by $a_{k-r+1}, a_{k-r+2}, \cdots, a_{k+1}, a_k$.

Step 2: When the input data shifts r+1 character to right, the divider system outputs $a_k b_r^{-1}$, which is the first coefficient x^{k-r}, at the same time, $a_k b_r^{-1}$ is transmitted into registers behind.

Step 3: According to the above steps, the processes are repeated until the divider system right shift k character. And the value of registers is the remainder.

The process of multiplier and divider combine the two processes above. The following flow chart is about $A(x)D(x) / B(x)$, which is shown as Fig. 3.

RS code encoding. According to multiplier and divider that we design above, we design a circuit of RS codes encoding. This paper denotes input data $g(x) = g_{2t}x^{2t} + g_{2t-1}x^{2t-1} + \ldots + g_1 x + g_0$ ($g_{2t=1}$), $m(x)$ $xn^{-k} = m_k x^{n-1} + m_{k-1}x^{n-2} + m_k x^{n-1} + \ldots + m_1 x^{n-k} + 0x^{n-k-1} + \ldots + 0\ xs + 0$. And the output of $m(x)$ xn^{-k}/ $g(x)$ is just the result of encoding. The circuit about $m(x)$ xn^{-k}/ $g(x)$ can be designed through multiplier and divider.

In practical application, $m(x)\ xn^{-k}$ is always stored in registers in order to decrease calculated amount. If so, all the process only need to carry out division and right shift n times.

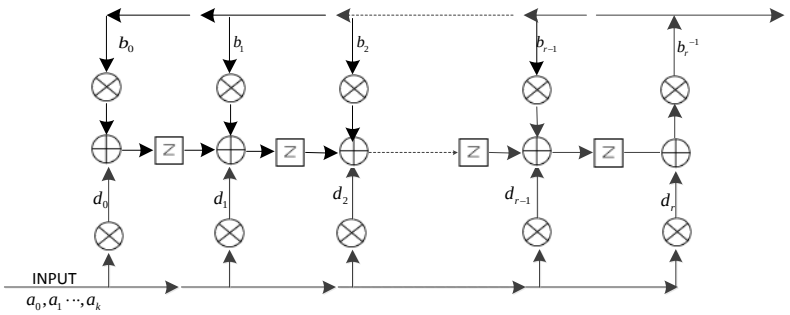

Fig. 3 The process of a multiplier and divider

The principle of RS code decoding

The process of decoding of RS code is the process that is to find the transmitted codes $c(x)$ from the received codes $r(x)$. And some errors may have been mixed into $r(x)$ because of disturbance and others. We called these errors error pattern and denoted them as $E(x)$.

Step 1. Figure out adjoint polynomial. Figure out adjoint polynomial S_j using received $r(x)$ as codes. As we all know, there are equations that reveal basic rules about generated matrix S and test matrix H and received vector R.

$$S^T = HR^T = \begin{bmatrix} a^{n-1} & a^{n-2} & \cdots & a & 1 \\ (a^2)^{n-1} & (a^2)^{n-2} & \cdots & a^2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ (a^{2t})^{n-1} & (a^{2t})^{n-2} & \cdots & a^{2t} & 1 \end{bmatrix} \begin{bmatrix} c_{n-1} \\ \vdots \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{2t} \end{bmatrix}$$

$$H = \begin{bmatrix} a^{n-1} & a^{n-2} & \cdots & a & 1 \\ (a^2)^{n-1} & (a^2)^{n-2} & \cdots & a^2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ (a^{2t})^{n-1} & (a^{2t})^{n-2} & \cdots & a^{2t} & 1 \end{bmatrix} \qquad S = RH^T$$

Where $R(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0$.

$x = \alpha$, $x = \alpha^2 \ldots x = \alpha^{2t}$ are respectively plugged in $R(x)$ to get the result of S_1, $S_2 \ldots S_{2t}$ .And we'll use that to get $S = RH^T = \left[R(\alpha), R(\alpha^2), \cdots, R(\alpha^{2t})\right]$ or $S_j = R(\alpha^j)$ which can be expressed as.

Step 2. Figure out the error position polynomial. We define

$$\Lambda(x) = (1 - X_1 x)(1 - X_2 x) \cdots (1 - X_v x)$$

$$= \prod_{l=1}^{v} (1 - X_l x) = \Lambda_v x^v + \Lambda_{v-1} x^{v-1} + \cdots + \Lambda_1 x + \Lambda_0$$

Where $1/X_1, 1/X_2, \cdots 1/X_v$ are roots of formula above and $X_1, X_2, \cdots X_v$ is the error position of expression.

In this case, through establishing 2t simultaneous equations, 2t roots of them can be obtained. These coefficients can be used to form Λ , and then we can figure out $\Lambda(x)$ using adjoin polynomial S quickly and easily, by applying Berlekamp-Massey algorithm to the problem. The algorithm is implemented according to the three following steps:

Firstly, we initialize

$$\Lambda_{-1}(x) = 1, D(-1) = 0, d_{-1} = 1$$

$$\Lambda_0(x) = 1, D(0) = 0, d_0 = s_1$$

We define that d_j is the interval between line j+1 and line j, and $D(j)$ is the order of $\Lambda_j(x)$.

Secondly, according to $d_j = s_{j+1} + \sum_{i=1}^{\partial+\Lambda_i(x)} s_{j+1-i}\Lambda_i(x)$, d_j is calculated. If $d_j = 0$, then $\Lambda_{j+1}(x) = \Lambda_j(x), D^*(j+1) = D^*(j)$. And iteration goes on while d_{j+1} need to be calculated. If $d_j \neq 0$, we should get line *i* ahead of line *j* so that $i - D(i)$ is maximum in all lines ahead of line j. According to $\Lambda_{j+1}(x) = \Lambda_j(x) - d_j d_i^{-1} x^{j-i}\Lambda^{(i)}(x)$, $\Lambda_{j+1}(x)$ is figured out.

Thirdly, we calculate d_{j+1} and repeat the second step. Through 2*t* iteration, $\Lambda_{2t}(x)$ can be figured out, and $\Lambda_{2t}(x)$ is just a result that needs to be evaluated.

Step 3. Figure out the root of error position polynomial. As we all know, roots of error position polynomials are the error positions. A method, which is called Chien search, can help us solve this problem effectively. For example, by Chien search, we test whether error exists in x^0 or not, and $x = 1/\alpha^0 = \alpha^n (\alpha^n = 1)$ is plugged into $\Lambda(x)$ formula, if the result is zero, we conclude that error exists in x^0, otherwise, it is right. The next process is to be implemented as the same way. $\alpha^{n-1}, \alpha^{n-2}, \cdots \alpha$ is respectively plugged into $\Lambda(x)$ to test $x^1, x^2, \cdots x^{n-1}$.

Step4. Figure out the error pattern. We have defined adjoint polynomial $S(x) = s_1 + s_2 x + s_3 x^2 + \cdots + s_{2t} x^{2t-1}$

And error position polynomial

$$\Lambda(x) = \prod_{l=1}^{v}(1 - X_l x) = \Lambda_v x^v + \Lambda_{v-1} x^{v-1} + \cdots + \Lambda_1 x + \Lambda_0$$

And we know $\Lambda_0 = 1$, if $x = X_l^{-1}$ then $\Lambda(x) = 0$. A polynomial is used to calculate the value of error, which is expressed as $\Omega(x) = S(x)\Lambda(x) (\mathrm{mod}\, x^{2t})$. According to Forney's algorithm, the effect of computing modulo x^{2t} is to discard all terms of degree 2*t* or higher. Therefore, we can get the error pattern expression of e_{ik}.

$e_{ik} = -\dfrac{\Omega(X_k^{-1})}{\Lambda'(X_k^{-1})}$ Where $\Lambda'(x)$ is the derivative of $\Lambda(x)$.

The final step is to add $\Lambda(x)$ with the received codes on error position in Galois field, and the sum is the result of decoding.

The result of simulation experiment and summary

We choose (*31, 25*) RS code as example to implement the whole simulation process of encoding and decoding. Its primitive polynomial is $x^5 + x^2 + 1$, which is obtained by look-up table or by using directives of rsgenpoly on MATLAB, and its signs take from $GF(2^5)$ with *m*=5.

The system is shown as Fig. 4.

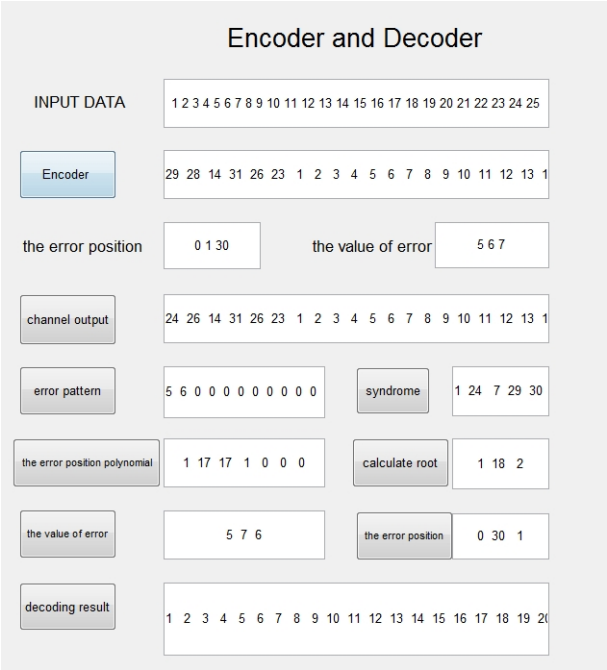

Fig. 4 The RS code encoder and decoder system

From the simulation experiment, we can conclude that the result of decoder is accurate by comparing decoding output characters and input characters. Of course, the number of error cannot more than 3, for the capability of error correcting is $(n\text{-}k)/2$ as to (n, k) RS.

According to the function of Profile Summary in MATLAB, we can analyze that there are 123 additions and 120 multiplications in whole process. It costs about 0.169s, so the system is successful to improve operation performance.

References

[1] C.E Shannon. A Mathematical Theory of Communication [J]. The Bell System Technical Journal, 27: p. 379-423(part I), p. 623-656(part II), Jul, Oct.1948.

[2] W. J. Gross, F. R. Kschischang, R. Koetter, P. G. Gulak. Simulation Results for Algebraic Soft-Decision Decoding of Reed-Solomon Codes [J]. Proc. of the 21st Biennial Symposium on Common. Kingston: p. 356-360, Jun. 2002.

[3] Y. WU, R. Koetter, C. Hadjicostic. Soft-decision list decoding of Reed-Solomon codes [J]. IEEE Common. Lett, 24(3): p. 481-490, Mar. 2006.

[4] Zhang Yong-guang, Lou Cai-yi. Channel coding and recognition technology [M]. Beijing: Publishing House of Electronics Industry, 2010.

[5] V. Guruswami, M. Sudan. Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes [J]. IEEE Trans. Inform. Theory, 45(6): p. 1757-1767, Sep. 1999.

[6] Error Correction Method for Reed-Solomon Decoding. US485-09 9-A. p. 258-261.