High-speed hardware decoder for double-error-correcting binary BCH codes

Shyue-Win Wei
Che-Ho Wei

Indexing terms: Algorithms, Codes and decoding, Errors and error analysis

Abstract: The paper presents a new hardware decoder for double-error-correcting binary BCH codes of primitive length, based on a modified step-by-step decoding algorithm. This decoding algorithm can be easily implemented with VLSI circuits. As the clock rate of the decoder is independent of block length and is only twice the data rate, the decoder is suitable for long block codes working at high data rates. The decoder comprises a syndrome calculation circuit, a comparison circuit and a decision circuit, which can be realised by linear feedback shift registers, ROMs and logical gates. The decoding algorithm, circuit design and data processing sequence are described in detail. The circuit complexity, decoding speed and data rate of the new decoder are also discussed and compared with other decoding methods.

1 Introduction

The Bose–Chaudhuri–Hocquenghem (BCH) codes are a class of extensively studied random-error-correcting cyclic codes [1–4]. A double-error-correcting binary BCH code of primitive length is capable of correcting any combination of two or fewer errors. This code is defined as follows [2]:

$$\text{block length} = n = 2^m - 1, \quad m \geqslant 3 \text{ (integer)}$$
$$\text{number of information bits} = k = n - 2m$$
$$\text{minimum distance} = d_{min} = 5$$

The generator polynomial of this code is specified in terms of its roots from the Galois field $GF(2^m)$. If α is a primitive element in the Galois field $GF(2^m)$, the generator polynomial $g(x)$ is the lowest degree polynomial over $GF(2)$ which has $\alpha^1, \alpha^2, \ldots, \alpha^4$ as its roots. Let $M_i(x)$ be the minimal polynomial of α^i, then $g(x)$ has been shown to be

$$g(x) = M_1(x)M_3(x) \tag{1}$$

and the degree of $g(x)$ is just $2m$. A summary of these BCH codes is given in Table 1 [2]. The popularly employed error-correcting procedure for double-error-correcting binary BCH codes consists of three major steps [1–4]:

(*a*) Calculate the syndrome values $S_i(i = 1, 2, 3)$ from the received word.

(*b*) Determine the error location polynomial $\sigma(x) = 1 + S_1 x + (S_2 + S_3/S_1)x^2$ from the syndrome values.

(*c*) Find the roots of $\sigma(x)$, which are the error locators.

Since the calculations in step (*a*) and step (*b*) in decoding the double-error-correcting binary BCH codes are quite simple, the work of step (*c*) becomes an important subject for decoding this kind of code. Chien's search algorithm is the most efficient. To determine the roots of $\sigma(x)$, many methods can be used to implement Chien's search algorithm [5–8]. The most straightforward is the lookup table method, although it is unpractical for long block codes [8]. Another hardware circuit which can be used to implement Chien's search algorithm in a straightforward manner is called the Chien searcher [1, 5]. This circuit needs two multipliers to compute $\sigma(x = \alpha^j)$. For long block codes, the circuit complexity increases and the cost of fast hradware circuits would be very expensive. Usually, a microprocessor-based software decoder can be used for long block codes [6–8]. However, due to the limited speed of central processing units, if a higher data rate is specified, a microprocessor-based method can only be used for medium block lengths (e.g. $m = 7$). Another algebraic decoding method, known as the step-by-step decoding method, involves changing the received symbols one at a time and testing to determine whether the weight of the error pattern has been reduced [3, 9–11]. The difference between this algorithm and Chien's search algorithm is that the step-by-step algorithm checks every potential error-location directly instead of searching the error-location numbers. A completely described step-by-step decoding algorithm for general BCH codes has been proposed by Massey [9], and also by Szwaja [10]. This general decoding algorithm has not been widely employed for large multiple-error-correcting codes owing to its requirement for calculation of the determinant of the syndrome matrix.

In this paper, a simple double-error-correcting decoder using a modification of the conventional step-by-step decoding method is proposed. It is suitable for high data rates and long block lengths of double-error-correcting binary BCH codes. Before introducing this modified step-by-step decoding algorithm, some properties of double-error-correcting binary BCH codes are briefly described in the following Section.

2 Properties of double-error-correcting binary BCH codes

If $K(x)$ is the $k - 1$ degree information polynomial, then the encoded codeword $c(x)$ can be expressed in a systematic form as

$$c(x) = K(x) + \text{mod } \{K(x)x^{n-k}/g(x)\}$$
$$= c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \tag{2}$$

Paper 6464I (E16), first received 3rd May and in revised form 21st October 1988

The authors are with the Institute of Electronics, National Chiao Tung University, 45 Po-Al Street, Hsin Chu, Taiwan, Republic of China

where, mod $\{K(x)x^{n-k}/g(x)\}$ indicates the remainder polynomial of $K(x)x^{n-k}$ divided by $g(x)$. Hereinafter, all codewords are assumed to be in systematic form. Let the error polynomial be $e(x)$, then the received-word polynomial is given by

$$\begin{aligned} r(x) &= c(x) + e(x) \\ &= r_0 + r_1 x + \cdots + r_{n-1}x^{n-1} \end{aligned} \tag{3}$$

If the syndrome values S_1 and S_3 are expressed in polynomial form as

$$S_i(x) = S_{i,0} + S_{i,1}x + \cdots + S_{i,m-1}x^{m-1}, \quad i = 1, 3 \tag{4}$$

then, by defining a remainder polynomial $b(x) = \text{mod}\,\{r(x)/M_i(x)\}$, the syndrome values can be also obtained from

$$S_i(x)\bigg|_{x=\alpha} = b(x)\bigg|_{x=\alpha^i}, \quad i = 1, 3 \tag{5}$$

Therefore, the calculation of syndrome values can easily be implemented by employing two m-stage shift registers with feedback connections [12], which are denoted as m-stage remainder shift register dividers. To correct all patterns of two or fewer random errors, the following relations between syndrome values can be applied [1]: if there is no error, then

$$S_1 = S_3 = 0 \tag{6a}$$

If there is one error only, then

$$S_1 \neq 0 \quad \text{and} \quad S_3 = (S_1)^3 \tag{6b}$$

If there are two errors, then

$$S_1 \neq 0 \quad \text{and} \quad S_3 \neq (S_1)^3 \tag{6c}$$

If there are three errors, the syndrome values could be

$$S_1 = 0 \quad \text{and} \quad S_3 \neq (S_1)^3, \qquad \text{or} \quad S_1 \neq 0 \quad \text{and} \quad S_3 \neq (S_1)^3 \tag{6d}$$

This is because the d_{min} of the codeword is equal to 5; the syndrome values could be the same as another codeword with two errors.

The cyclic structure of BCH codes was proved by Peterson in 1960 [12]. If $r^{(j)}(x)$ is obtained by cyclically shifting j bits of $r(x)$ to the right, then it can be expressed as

$$\begin{aligned} r^{(j)}(x) &= \text{mod}\left\{\frac{x^j r(x)}{x^n + 1}\right\} \\ &= r_{n-j} + r_{n-j+1}x + \cdots + r_{n-1}x^{j-1} \\ &\quad + r_0 x^j + \cdots + r_{n-j-1}x^{n-1} \end{aligned} \tag{7}$$

Similarly, the shifted remainder polynomials $b_1^{(j)}(x)$ and $b_3^{(j)}(x)$ can be obtained as

$$\begin{aligned} b_i^{(j)}(x) &= \text{mod}\,\{r^{(j)}(x)/M_i(x)\} \\ &= \text{mod}\,\{x^j b_i(x)/M_i(x)\}, \quad i = 1, 3 \end{aligned} \tag{8}$$

Hence $b_i^{(j)}(x)$ can be obtained by cyclically shifting $b_i(x)$ by j bits and dividing by $M_i(x)$ (that is, $b_i^{(j)}(x)$ can be obtained by shifting j bits in the remainder shift-register dividers). The corresponding syndrome values are then given by

$$S_i^j(\alpha) = b_i^{(j)}(\alpha^i), \quad i = 1, 3 \tag{9}$$

Suppose that it is known that one error occurs at location x^{n-j}. Let $e_1(x) = x^{n-j}$ and $r(x) = r_1(x) + e_1(x)$. After shifting the polynomial $r(x)$ j bits to the right, the new error pattern becomes $e_1^{(j)}(x) = 1$, and the shifted remainder polynomials $b_i^{(j)}(x)$ are found to be

$$\begin{aligned} b_i^{(j)}(x) &= \text{mod}\,\{r^{(j)}(x)/M_i(x)\} \\ &= \text{mod}\,\{r_1^{(j)}(x)/M_i(x)\} + 1 \\ &= \overline{b_i^{(j)}(x)} + 1, \quad i = 1, 3 \end{aligned} \tag{10}$$

For binary BCH codes, eqn. 10 can be rewritten as

$$\overline{b_i^{(j)}(x)} = b_i^{(j)}(x) + 1, \quad i = 1, 3 \tag{11}$$

Here, $r_1(x)$ implies the received-word polynomial after correcting the x^{n-j} location's error, and the polynomial $\overline{b_i^{(j)}(x)}$ is the new remainder polynomial. Consequently, the corresponding new syndrome values are given by

$$\overline{S_i^j(\alpha)} = \overline{b_i^{(j)}(\alpha^i)}, \quad i = 1, 3 \tag{12}$$

3 Modified step-by-step decoding algorithm

The principle of the conventional step-by-step decoding algorithm can be summarised as three steps [9–10]. First, it determines the original order of the syndrome matrix (i.e. the number of errors that occur in the received word) by using an iteration method. The syndrome matrix is defined in property 4′ of Reference 9. Secondly, it temporarily changes the syndrome values and then corrects the error if the changed syndrome matrix is singular. Finally, it decreases the order of the syndrome matrix by one if an error is found, and shifts one bit to repeat the second step. A new method, based on the properties described above, is presented here. In this method the syndrome values are first changed and then the error in terms of the difference between the initial syndrome values and new syndrome values is corrected. This idea is based on the fact that the number of errors can be determined in terms of the patterns of syndrome values as shown in eqns. 6. This modified algorithm has two advantages:

(*a*) it avoids the iteration loop in step 1 of the conventional algorithm

(*b*) it can be easily implemented by hardware circuits.

The flowchart of this modified decoding algorithm is illustrated in Fig. 1.

Using eqn. 5 one obtains the initial syndrome values S_1^0 and S_3^0 corresponding to the received word polynomial $r(x)$ before any correction. The number of error bits existing in the received word $r(x)$ can be determined from eqns. 6. If three or more error bits are detected $[S_1 = 0$ and $S_3 \neq (S_1)^3]$, then the next operation is unnecessary and an alarm or ARQ signal is sent back to end the decoding procedure; otherwise, the next decoding steps are implemented.

The decoder begins to implement its checking algorithm after the initial syndrome values have been obtained. First, assume that the x^{n-1} position of the received polynomial $r(x)$ corresponds to an error bit. Then, shifting $r(x)$ one bit to the right will make the error bit occur at position x^0. Meanwhile, the contents of the two remainder shift-register dividers are also shifted. To correct the x^{n-1} location error [i.e. x^0 position of $r^{(1)}(x)$] in the syndrome values, the remainder shift-register dividers are incremented by '1' using eqn. 11. Hence the corresponding error-corrected remainder polynomials $\overline{b_i^{(1)}(x)}$ and syndrome values $\overline{S_1^1}$, $\overline{S_3^1}$ can be determined. Finally, eqn. 6 is used to check the syndrome values obtained. If the assumption is true, then the first error bit has been found and the number of error bits will decrement by one. If the assumption is false, then an extra error is added to the remainder polynomial and syndrome values, so that the number of error bits will increment by one. Thus, in terms of the change in error bit number one can determine whether x^{n-1} position of $r(x)$ is an error bit or not. When the error bit is detected, the decoder adds correcting bit $E_c = 1$ at the x^{n-1} location of $r(x)$; otherwise, an erasure bit $\bar{E}_c = 1$ is added to the remainder shift-register dividers to erase the effect of the error-bit assumption. After decoding the x^{n-1} location of $r(x)$, the decoder proceeds to decode the x^{n-2} position of $r(x)$ by repeating the above procedure. When k information bits have been checked and corrected, the received word is decoded completely. Here, the period for decoding one information bit is defined as a 'cycle'.

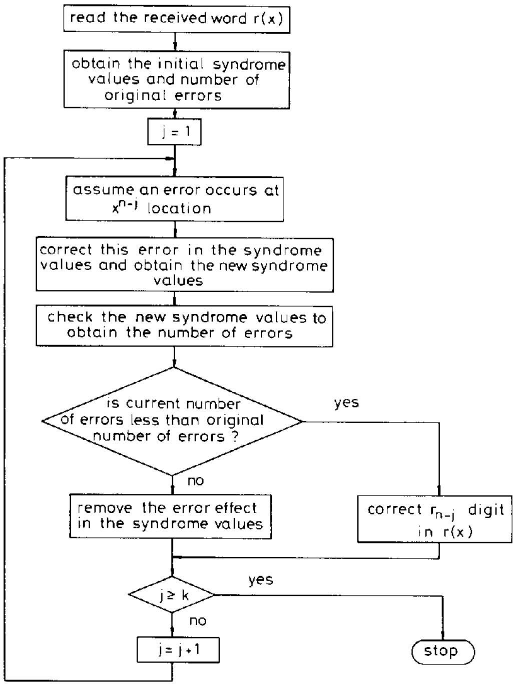
read the received word r(x)
obtain the initial syndrome values and number of original errors
j = 1
assume an error occurs at x^{n-j} location
correct this error in the syndrome values and obtain the new syndrome values
check the new syndrome values to obtain the number of errors
is current number of errors less than original number of errors ?
yes
no
remove the error effect in the syndrome values
correct r_{n-j} digit in r(x)
j ≥ k
yes
no
j = j + 1
stop

Fig. 1 *Modified step-by-step decoding algorithm*

4 Hardware decoder

The modified step-by-step decoding algorithm can be implemented by a simple structure and with simple hardware circuits. Fig. 2 shows the functional block diagram of this decoder. It is partitioned into three parts: syndrome calculation circuit, comparison circuit and decision circuit. The syndrome calculation circuit is used to obtain the new syndrome values $\bar{S}_1$ and $\bar{S}_3$. The comparison circuit is used to determine whether or not $S_1 = 0$ and $S_3 = (S_1)^3$. The decision circuit is used to check whether the error-bit assumption is true or false. After checking, if the error-bit assumption is true, the correcting bit E_c should be added to the readout information bit; otherwise, an erasure bit $\bar{E}_c$ is fed back to the syndrome calculation circuit to erase the effect of the error assumption. The detailed design of these three circuits is described in the following.

(*a*) *Syndrome calculation circuit:* The remainder polynomials $b_1^{(j)}(x)$ and $b_3^{(j)}(x)$ should add '1' if the error-bit assumption is true; an erasure bit $\bar{E}_c$ is added if the error-bit assumption is false. Thus, this circuit can be designed by slightly modifying the conventional syndrome calculation circuit [2], that is a '1' is added to the inputs of first stages of the remainder shift-register dividers in the decoding procedure and an erasure bit $\bar{E}_c$ is added to the outputs of first stages of the remainder shift-register dividers.

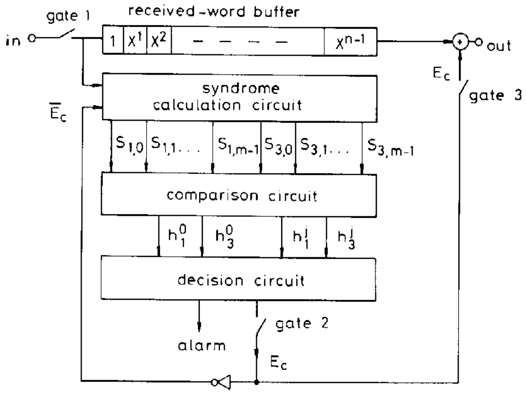
gate 1
received-word buffer
in
1
x^1
x^2
x^{n-1}
out
E_c
gate 3
$\bar{E}_c$
syndrome calculation circuit
$S_{1,0}$ $S_{1,1}$ ... $S_{1,m-1}$ $S_{3,0}$ $S_{3,1}$ ... $S_{3,m-1}$
comparison circuit
h_1^0 h_3^0 h_1^j h_3^j
decision circuit
gate 2
alarm
$\bar{E}_c$

Fig. 2 *Functional block diagram of the new decoder*

(*b*) *Comparison circuit:* To determine the number of error bits in eqn. 6, two comparisons must be performed:

(*i*) Is S_1 equal to zero?

(*ii*) Is S_3 equal to $(S_1)^3$?

The results of these two comparisons can be represented by two bits. These two new variables h_1 and h_3 are called 'checking bits':

$$\text{If } S_1^j(\alpha) = 0, \quad \text{then } h_1^j = 1 \qquad (13a)$$

$$\text{If } S_3^j(\alpha) = \{S_1^j(\alpha)\}^3, \quad \text{then } h_3^j = 1 \qquad (13b)$$

The corresponding circuit is shown in Fig. 3. Since $(S_1^j)^3$ should be modulo $M_1(x)$ to a polynomial of degree $m - 1$ or less, the cubic operation on $S_1^j(\alpha)$ in eqn. 13*b* can be implemented by a ROM of size $2^m \times m$ bits using the lookup table method.

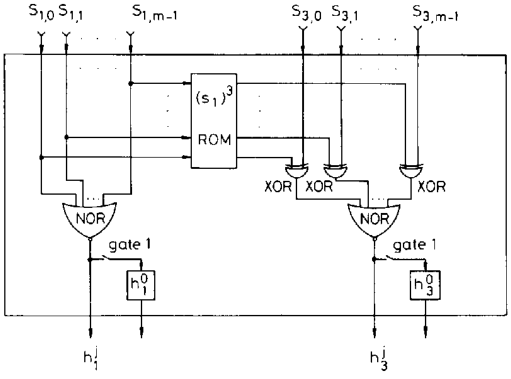

Fig. 3 *Comparison circuit*

(*c*) *Decision circuit:* To check whether or not the information bit in a current cycle is an error bit, four parameters h_0^0, h_3^0, h_1^j, h_3^j are employed. Here h_1^0 and h_3^0 are the initial values which represent the original number of error bits in the received word before decoding, while h_1^j and h_3^j represent the number of error bits in the current decoding cycle. If the number of error bits in the current cycle is less than the initial value (i.e. the original number of error bits), then the corresponding bit must be an error bit and the decoder sends a correcting bit $E_c = 1$ to the output of the received word buffer. Therefore, the decision circuit can be implemented by a ROM of size $2^4 \times 2$ bits or by a logic circuit as shown in Fig. 4.

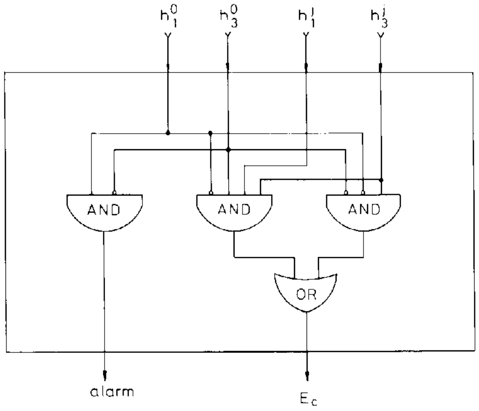

Fig. 4 *Decision circuit*

The proper operation of this decoder can be summarised in the following steps:

(1) For $j = 0$, gate 1 and gate 3 switch on while gate 2 switches off; the received word vector is read into the buffer with the high-order bits first. If we use the same cycle time to do the shifting operations, then this step needs n cycles. After the nth cycle, initial values (b_1^0, b_3^0), (S_1^0, S_3^0) and (h_1^0, h_3^0) are obtained. If $h_1^0 = 1$ and $h_3^0 = 0$, then three or more error bits are detected and the decoder returns an alarm or ARQ signal; otherwise, the decoder proceeds to step 2.

(2) For $j = 1$, gate 2 switches on and gate 1 switches off.

(3) Cyclically shift one bit right for the remainder shift register dividers to get $(\overline{b_1^{(j)}(x)}, \overline{b_3^{(j)}(x)})$, $(\overline{S_1^j}, \overline{S_3^j})$ and (h_1^j, h_3^j). After the decision operation, the corresponding correcting bit E_c and erasure bit $\bar{E}_c$ can be obtained. Now, the error-correcting bit E_c obtained will be added to bit r_{n-j} when it leaves the received word buffer. The erasure bit $\bar{E}_c$ is ready to be added in the remainder shift-register dividers during the next cycle.

(4) If all errors are corrected (i.e $h_1^j = 1$ and $h_3^j = 1$), then gate 3 switches off at the beginning of the next cycle.

(5) If $j = k$, then the procedure is stopped (two LFSRs are cleared ready for decoding the next received word); otherwise, j is increased by one and the operation proceeds to step 3.

In step 1 of a complete decoding period, the first $n - 1$ clock cycles are simple shifting operations. In the last cycle of step 1 and in k cycles in step 3, more complicated operations should be completed. Therefore, the decoding time for the last $k + 1$ clock cycles will be longer than for the first $n - 1$ cycles. The data processing sequence in one complete decoding cycle is illustrated in Fig. 5. Also, it should be noted that the operation for 'error-bit assumption' of the jth cycle and the erasure of the effect of false 'error-bit assumption' of the $(j - 1)$th cycle can be processed at the same instant.

5 Hardware complexity and data rate

The regular structure of this decoder reduces the hardware complexity and makes it easy to design. For a given desired block length (i.e. given m), the length of the received word buffer and the stage number of the remainder shift register dividers can be determined first. Then the syndrome calculation circuit can be designed using $M_1(x)$ and $M_3(x)$. Next, the $2^m \times m$-bit ROM in the comparison circuit is implemented using polynomial $M_1(x)$. Finally, since the decision circuit is independent of block length, it is fixed for any given m. In general, the new decoder can easily be implemented for block lengths up to $2^{10} - 1$. The information required to design this decoder is given in Table 1. That this decoder is easier to implement than the conventional step-by-step decoder is clearly seen by comparing Fig. 2 and Fig. 1 of Reference 9. Also, this decoder is faster and has lower circuit complexity than the hardware decoder of the Chien searcher [1, pp. 132–136; 5, Fig. 2]. This is because the Chien searcher requires two multipliers to compute $\sigma(x = \alpha^j)$, and the cost of building wired net to multiply by α^2 in $GF(2^m)$ in one clock period becomes substantial for larger m. Usually, it is more economical to allow two shift operations or m clock periods for the multiplication of α^2; thus, the decoding speed is degraded.

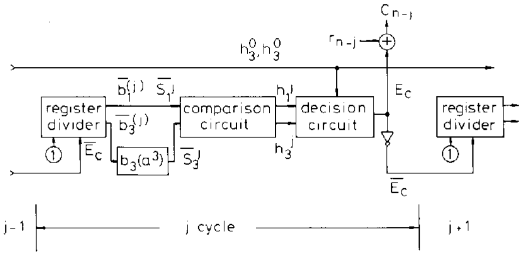

Fig. 5 *Data processing sequence in one cycle*

Table 1: Double-error-correcting binary BCH codes of primitive length

m	Code rate k/n	Minimal polynomials	
		$M_{1,0} \cdots M_{1,m-1}$	$M_{3,0} \cdots M_{3,m-1}$
3	0.14	110	101
4	0.47	1100	1111
5	0.68	10100	10111
6	0.81	110000	111010
7	0.89	1001000	1111000
8	0.94	10111000	11101110
9	0.96	100010000	100110100
10	0.98	1001000000	1111000000

$M_1(x) = 1 + M_{1,1}x + M_{1,2}x^2 + \cdots + M_{1,m-1}x^{m-1} + x^m$
$M_3(x) = 1 + M_{3,1}x + M_{3,2}x^2 + \cdots + M_{3,m-1}x^{m-1} + x^m$

The new decoder is also faster and easier to implement than one using the microprocessor-based method [6–8]. Comparing this new decoder with the fast Chien's search method with respect to hardware complexity [8], we find that:

(a) a fast Chien's search algorithm needs a ROM of size $2^{2m} \times 1$ bit as the segment identifier table, which is 16384 bits for $m = 7$. The new decoder, however, contains a ROM of size $2^m \times m$ bits in the comparison circuit, which is only 896 bits for $m = 7$

(*b*) the syndrome calculation circuits for these two algorithms are comparable

(*c*) the fast Chien's search algorithm needs a 16-bit processor whereas the new decoder needs only $m + 6$ logic gates.

Decoding speed (or data rate) is another important factor, which is usually used to estimate decoder efficiency. When the fast Chien's search algorithm is employed with the microprocessor-based method, the data rate is 383 kbit/s for $m = 7$ if the clock rate of the processor is 10 MHz. In the new decoder, if f_c is the clock rate, then the data rate is $(n/n + k)f_c$ bit/s, which is 1066 kbit/s for a 2 MHz clock rate and $m = 7$. Even when the word length is very long, the decoder can keep the data rate to at least half of the clock rate. The clock rate and decoding speed depend mainly on the access time of the ROM. At the time of writing most commercial ROMs have an access time in the range 150–250 ns. Specially designed high-speed ROMs with access times under 35 ns and using flash EEPROM technology are available also (e.g. the XL46HC64 SpeedPROM).

6 Conclusions

A new decoder based on a modified step-by-step decoding algorithm for double-error-correcting binary BCH codes has been presented, which requires only n clock cycles for reading the received word and k clock cycles for finding the error bits. Since the checking operation of this modified algorithm only needs to compare the number of current errors with the original errors, the comparison circuit and decision circuit are simplified. Also, the properties of BCH codes described in Section 2 make the error-bit assumption very easy to implement. The working data rate of the decoder depends only on the clock rate, which is determined mainly by the access time of the ROM in the comparison circuit and is independent of the block length. Therefore, this new decoder may work at a high data rate for high rate codes. Because of its simplicity in structure and circuit realisation, this decoder may easily be implemented using VLSI circuits. If the data rate is specified, the clock generator can be integrated with the decoder and encoder in one chip.

The decoding algorithm and circuit structure can be extended for complete decoding of double-error-correcting binary BCH codes. In Reference 1 (Section 16.48), it is shown that if

$$Tr\left[\frac{S_3 + (S_1)^3}{(S_1)^3}\right] = 1$$

where

$$Tr[x] = \sum_{i=0}^{m-1} x^{2^i}$$

then three or more errors are detected. Thus, the decoder is a complete decoder for double-error-correcting binary BCH codes if provision is made in the comparison circuit and decision circuit:

(*a*) to create a new check bit, say h_0, to indicate the value of trace. Here, the calculation of the trace can be straightforwardly implemented by a ROM of size $2^{2m} \times 1$ bits

(*b*) to refresh the fixed initial values h_i^0 in the bounded-distance decoder by h_i^j (for $i = 0, 1, 3$) when an error bit is found (i.e. $h_i^0 = h_i^j$ if $E_c = 1$).

Similarly, the decoding algorithm and circuit structure can also be extended for other binary BCH codes.

7 Acknowledgment

The work reported in this paper was supported by the National Science Council of the Republic of China as research project NSC 78-0404-E009-09.

8 References

1 BERLEKAMP, E.R.: 'Algebraic coding theory' (Aegean Park Press, 1984)

2 LIN, S., and LOSTELLO, D.J., Jr.: 'Error control coding' (Prentice-Hall, 1983)

3 PETERSON, W.W., and WELDON, E.J., Jr.: 'Error-correcting codes' (MIT Press, Cambridge, MA, 1972)

4 MICHELSON, A.M., and LEVESQUE, A.H.: 'Error-control techniques for digital communication' (John Wiley & Sons, Inc., 1985)

5 CHIEN, R.T.: 'Cyclic decoding procedures for Bose–Chaudhuri–Hocquenghem codes', *IEEE Trans.*, 1964, **IT-10**, pp. 357–363

6 BARTEE, T.C., and SCHNEIDER, D.I.: 'An electronic decoder for Bose–Chaudhuri–Hocquenghem error-correcting codes', *IRE Trans.*, 1962, **IT-8**, pp. S17–24

7 LE-NGOC, T., and BHARGAVA, V.K.: 'A microprocessor based decoder for BCH codes', *Can. Electr. Eng. J.*, 1979, **4**, pp. 29–32

8 SHAYAN, Y.R., LE-NGOC, T., and BHARGAVA, V.K.: 'Binary-decision approach to fast Chien search for software decoding of BCH codes', *IEE Proc. F, Commun., Signal & Process.*, 1987, **134**, (6), pp. 629–632

9 MASSEY, J.L.: 'Step-by-step decoding of the Bose–Chaudhuri–Hocquenghem codes', *IEEE Trans.*, 1965, **IT-11**, pp. 580–585

10 SZWAJA, Z.: 'On step-by-step decoding of the BCH binary code', *IEEE Trans.*, 1967, **IT-13**, pp. 350–351

11 HARTMANN, C.R.P.: 'A note on the decoding of double-error-correcting binary BCH codes of primitive length', *IEEE Trans.*, 1971, **IT-17**, pp. 765–766

12 PETERSON, W.W.: 'Encoding and error-correction procedures for the Bose–Chaudhuri codes', *IRE Trans.*, 1960, **IT-6**, pp. 459–470