A High-Speed Real-Time Binary BCH Decoder

Shyue-Win Wei, *Member, IEEE*, and Che-Ho Wei, *Senior Member, IEEE*

***Abstract*—A high-speed real-time decoder for *t*-error-correcting binary Bose–Chaudhuri–Hocquenghem (BCH) codes based on a modified step-by-step decoding algorithm is presented. The average operation cycles for decoding each received word is just equal to the block length of the codeword. The decoder is constructed by three modules: the syndrome module, the comparison module, and the error corrector. Since all of the modules can be implemented by systolic circuits, the operation data rate of this decoder can theoretically be up to a rate of the inverse of two logic-gate delays. Based on different VLSI technologies, such as CMOS, BiCMOS and GaAs, the decoder can be operated from approximately several hundreds megabits per second to the order of gigabits per second. Thus, the decoder can be applied in the broadband service and video processing. Besides, by avoiding the use of inverse operation in the step-by-step decoding method, the circuit complexity of this decoder can be much less than the standard algebraic method in which the inverse operation is usually required for finding the coefficients of the error-location polynomial. The detailed circuit diagrams of the comparison module and error corrector for the double- and triple-error-correcting binary BCH codes are given for illustration.**

***Keywords*—BCH code; error-control coding; real-time implementation; VLSI architecture.**

I. Introduction

THE Bose–Chaudhuri–Hocquenghem (BCH) codes are a class of most extensively studied random-error-correcting cyclic codes [1]–[5]. The cyclic structure of BCH codes has been proved by Peterson in 1960 [6]. The most popular error-correcting procedure for the binary BCH codes is the standard algebraic decoding method consisting of three major steps [1]–[5]:

1) calculate the syndrome values S_i, $i = 1, 2, \cdots, 2t$ from the received-word polynomial $r(x)$;
2) determine the error-location polynomial $\sigma(x)$ from the syndrome values of the received word; and
3) solve for the roots of $\sigma(x)$, which are the error locators.

Among these decoding steps, the Berlekamp's iteration algorithm for Step 2 and Chien's search algorithm for Step 3 are the most efficient. Another algebraic decoding method, known as the step-by-step decoding method, was first presented by Massey in 1965 for the general cases of BCH codes [7]. The basic principle of the conventional step-by-step decoding method is that it involves changing received symbols one at a time with testing to determine whether the weight of error pattern has been reduced. The method is less complex than the standard algebraic method since the step-by-step method avoids calculating the coefficients of error-location polynomial and searching the roots [7]. Another major advantage of the step-by-step method in hardware implementation is that there is no need for inverse operation in the decoding process. To simplify hardware implementation, a modified step-by-step decoder for decoding the double-error-correcting binary BCH codes was recently presented by the authors [8]. The basic principle of the modified step-by-step decoding algorithm is that it directly compares the number of errors in the current cycle with that in the previous cycle. However, this step-by-step decoder is not a real-time decoder since it requires $n + k$ clock cycles for decoding one received word, where n is the block length and k is the length of information bits. In addition, the comparison circuit is designed using static read-only memory (ROM), thus the decoding speed of the decoder is limited by the computation time of the comparison module.

Using some results in [3]–[7], the idea presented in [8] can be extended for decoding a general t-error-correcting binary BCH code. Furthermore, a shifted-syndrome generator is added into the decoder to enable the decoder to decode consecutive input code words in real time. The matrix calculation circuit in the comparison module is now designed by systolic circuits, thus the average computation time for the high-speed real-time decoder is only two logic-gate delays.

For the video-signal transmission and broadband service, very high data rate of transmission is usually required. Based on current VLSI technologies, such as CMOS, BiCMOS [9], and GaAs [10], propagation delay of one logic gate can vary from nanoseconds (ns) to picoseconds (ps). It implies that the decoder can be operated from several hundred megabits per second up to gigabits per second if the average decoding time of each bit is only two logic-gate delays.

II. Binary BCH Codes

A t-error-correcting binary-primitive BCH code is designed to be capable of correcting any combination of t or

Manuscript received March 27, 1992; revised July 20, 1992. Paper was recommended by Associate Editor Peter Pirsch.

C.-H. Wei is with the Institute of Electronics and Center for Telecommunications Research, National Chiao Tung University, Hsinchu, Taiwan 300, Republic of China (author to whom correspondence should be addressed).

S.-W. Wei is with the Telecommunications Laboratories, Chung-Li 32099 Taiwan, Republic of China.

IEEE Log Number 9207314.

fewer errors and can be denoted as $(n, k, d_{\min})$ *bp* BCH code. The code is defined as follows [1]–[5]

Block length:	$n = 2^m - 1, m \geq 3$ (integer)
Number of information bits:	$k \geq n - mt$
Minimum distance:	$d_{\min} \geq 2t + 1$

The generator polynomial of the 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 $\alpha^1, \alpha^2, \cdots, \alpha^{2t}$ as its roots. Let (x) be the minimal polynomial of α^i, then $g(x)$ is the least common multiple (lcm) of $M_1(x), M_3(x), \cdots, M_{2t-1}(x)$, that is

$$g(x) \equiv \operatorname{lcm}\{M_1(x), M_3(x), \cdots, M_{2t-1}(x)\}. \quad (1)$$

The degree of each minimal polynomial is m or less, the degree of $g(x)$ is therefore at most mt. In fact, the degree of the generator polynomial is $2m$ for $t = 2$, and is $3m$ for $m > 4$ if $t = 3$.

The encoding process of a *bp* BCH code is the same as the typical cyclic code and can be described as

$$\begin{aligned} C(x) &= K(x)x^{n-k} + \operatorname{mod}\{K(x)x^{n-k}/g(x)\} \\ &= c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \end{aligned} \quad (2)$$

where $K(x)$ is the associated information polynomial and $\operatorname{mod}\{K(x)x^{n-k}/g(x)\}$ indicates the remainder polynomial of $K(x)x^{n-k}$ divided by $g(x)$. The encoding circuit for a systematic $(n, k, d_{\min})$ *bp* BCH code can be implemented by an $(n - k)$-stage linear-feedback-shift-register (LFSR) circuit [3].

Let $e(x)$ be an error polynomial and $C(x)$ be a systematic code-word; the received polynomial $r(x)$ can be expressed as

$$\begin{aligned} r(x) &= C(x) + e(x) \\ &= r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1}x^{n-1} \end{aligned} \quad (3)$$

and the corresponding syndrome values can be computed by

$$\begin{aligned} S_i^0(\alpha) &= r(x)_{|x=\alpha^i} \\ &= e(x)_{|x=\alpha^i} \\ &= \operatorname{mod}\{r(x)/M_i(x)\}_{|x=\alpha^i} \\ &= S_{i,0}^0 + S_{i,1}^0\alpha + S_{i,2}^0\alpha^2 + \cdots + S_{i,m-1}^0\alpha^{m-1} \\ & \qquad i = 1, 3, \cdots, 2t - 1 \end{aligned} \quad (4)$$

and $S_{2i}^0 = (S_i^0)^2$, $i = 2, 4, \cdots, 2t$ for the *bp* BCH codes [1]–[5]. In the paper, superscript "0" of S_i^0 means no shift operation of the received word is performed. S_i^0, $i = 1, 3, \cdots, 2t - 1$, are called *initial syndrome values* hereafter. Each syndrome value can be expressed as a polynomial of degree $m - 1$, or an m-tuple vector. In practice, the syndrome values can be computed by using a syndrome generator composed of t pieces of m-stage LFSR's [1]–[5]. Clearly, each $S_i(x) = e(x)$ if the degree of $e(x)$ is less than the degree of $M_i(x)$.

III. Decoding Algorithm

The basic principle of the step-by-step decoding method is that it involves changing the received bits one at a time by testing to determine whether the weight-of-error pattern has been reduced. Therefore, the relationship between syndrome and weight-of-error pattern should be determined first. For a t-error-correcting *bp* BCH code, the relations among syndrome values can be found by using Peterson's direct-solution method [2], property 4′ of [7], or theorem 9.11 of [3]. For consistency in the following presentation, the theorem is rewritten as follows:

Theorem 1: For an $(n, k, d_{\min})$ *bp* BCH code, let syndrome matrix $\mathbf{L}_p^0$, $1 \leq p \leq t$ be given by

$$\mathbf{L}_p^0 = \begin{bmatrix} S_1^0 & 1 & 0 & \cdots & 0 \\ S_3^0 & S_2^0 & S_1^0 & \cdots & 0 \\ & \vdots & & & \vdots \\ S_{2p-1}^0 & S_{2p-2}^0 & S_{2p-3}^0 & \cdots & S_p^0 \end{bmatrix}, \qquad p = 1, 2, \cdots, t.$$

Then, $\mathbf{L}_p^0$ is singular if the number of errors is $p - 1$ or less and is nonsingular if the number of errors is p or $p + 1$.

Using the theorem, the number of errors can be bounded in terms of $\det(\mathbf{L}_1^0), \det(\mathbf{L}_2^0), \cdots, \det(\mathbf{L}_t^0)$. For instance, $\det(\mathbf{L}_4^0) = 0$ implies that the number of errors is three or less. Furthermore, the number of errors can be determined in terms of the relations among $\det(\mathbf{L}_1^0)$, $\det(\mathbf{L}_2^0), \cdots, \det(\mathbf{L}_t^0)$. For example, if $\det(\mathbf{L}_1^0) \neq 0$, $\det(\mathbf{L}_2^0) \neq 0$, and $\det(\mathbf{L}_p^0) = 0$ for $p = 3, 4, \cdots, t$, then two errors have occurred. Since we only care whether or not the value of $\det(\mathbf{L}_p^0)$ is equal to zero, the results can be denoted by using t decision bits h_p^0 $(p = 1, 2, \cdots, t)$, defined by

$$h_p^0 = 1 \qquad \text{if } \det\left(\mathbf{L}_p^0\right) = 0, \qquad p = 1, 2, \cdots, t. \quad (5)$$

Using the decision bits, a decision vector $\mathbf{H}^0$ is defined as

$$\mathbf{H}^0 = (h_1^0, h_2^0, \cdots, h_t^0). \quad (6)$$

Thus, the number of errors can be uniquely determined in terms of the decision vector $\mathbf{H}^0$ if and only if the number of errors is t or less. From implementation point of view, the decision vector can be regarded as different determinants are computed in parallel. For example, if $t = 2$, it can be found that

- If there is no error, then $\mathbf{H}^0 = (1, 1)$.
- If there is one error, then $\mathbf{H}^0 = (0, 1)$.
- If there are two errors, then $\mathbf{H}^0 = (0, 0)$.

Using Theorem 1, the decision vector of a general t-error-correcting *bp* BCH code can be determined as follows:

- If there is no error, then $\mathbf{H}^0 \in \phi_0 = \{(1^t)\}$, where 1^t indicates t consecutive identical bits of 1. For example, vector $(1^3) = (1, 1, 1)$.

- If there is one error, then $\mathbf{H}^0 \in \phi_1 = \{(0, 1^{t-1})\}$
- If there are ζ errors, $2 \le \zeta < t$, then $\mathbf{H}^0 \in \phi_\zeta = \{(\mathrm{x}^{\zeta-2}, 0, 0, 1^{t-\zeta})\}$, where the symbol "x" can be "0" or "1."
- If there are t errors, then $\mathbf{H}^0 \in \phi_t = \{(\mathrm{x}^{t-2}, 0, 0)\}$.

In general, $\phi\zeta$ $(0 \le \zeta \le t)$ is a set of all possible decision vectors that ζ errors have occurred.

From the above rules, the decision vectors of various weights-of-error patterns can be distinguished from one another if the weights of the error patterns are t or less. Thus, the number of errors can be correctly determined in terms of the pattern-of-decision vector if and only if the weight-of-error pattern is t or less. Since the *bp* BCH codes are an important class of cyclic codes, the code words and the received words can be cyclically shifted without losing their information of syndrome. Using the cyclical properties of the *bp* BCH codes, if the first position of $r(x)$, r_{n-1}, can be decoded correctly for all correctable error patterns, then the entire word can be decoded correctly [3], [7]. If $r^j(x)$ is obtained by cyclically shifting $r(x)$ j places to the right, then it is known that the corresponding syndrome, denoted by S_i^j, can be obtained by shifting the contents of the LFSR's j times in a syndrome generator with initial contents [3, theorem 8.7].

Let us first denote that

$$\bar{S}_i^j = S_i^j, \qquad \text{for } j = 0; \quad i = 1, 3, \cdots, 2t-1 \tag{7a}$$

$$\bar{S}_i^j = S_i^j + 1, \qquad \text{for } 1 \le j \le n; \quad i = 1, 3, \cdots, 2t-1. \tag{7b}$$

S_i^0 and $S_i^j + 1$, $j = 1, 2, \cdots, n$; $i = 1, 3, \cdots, 2t-1$ in (7) are represented by a unified symbol where $\bar{S}_i^j$, where $\bar{S}_i^0 = S_i^0$ $(i = 1, 3, \cdots, 2t-1)$ are initial syndrome values of $r(x)$, and $\bar{S}_i^j$ $(j \ge 1;\ i = 1, 3, \cdots, 2t-1)$ are syndrome values of $r^j(x) + 1$. That the magnitude of the j bit place of $r(x)$, r_{n-j} is changed is indicated by $r^j(x) + 1$. Some corresponding decision bits can also be defined in the following:

$$h_p^j = 1 \qquad \text{if } \det\left(\mathbf{L}_p^j\right) = 0, \qquad p = 1, 2, \cdots, t; 1 \le j \le n \tag{8}$$

where

$$\mathbf{L}_p^j = \begin{bmatrix} \bar{S}_1^j & 1 & 0 & \cdots & 0 \\ \bar{S}_3^j & \bar{S}_2^j & \bar{S}_1^j & \cdots & 0 \\ & \vdots & & & \vdots \\ \bar{S}_{2p-1}^j & \bar{S}_{2p-2}^j & \bar{S}_{2p-3}^j & \cdots & \bar{S}_p^j \end{bmatrix},$$

$$p = 1, 2, \cdots, t; 1 \le j \le n.$$

Finally, these decision bits can be used to form a decision vector $\mathbf{H}^j$:

$$\mathbf{H}^j = (h_1^j, h_2^j, \cdots, h_t^j), \qquad 1 \le j \le n. \tag{9}$$

Thus, $\mathbf{H}^0$ is the decision vector of initial syndrome values and $\mathbf{H}^j$, $j \ge 1$ is the decision vector of temporarily changed syndrome values, $S_i^j + 1$, which indicate that the magnitude of the first position of $r^j(x)$ is temporarily changed. Thus, the number of errors represented by $\mathbf{H}^j$ will decrease by one if the first position of $r^j(x)$, r_{n-j} is an erroneous bit; otherwise, an extra error is added to $r^j(x)$ and the number of errors will increase by one. Obviously, the weight difference between the error vector represented by H^0 and the error vector represented by $\mathbf{H}^j$ is one. Thus, the first position of $r^j(x)$ can be determined to be an erroneous bit or not in terms of the difference between $\mathbf{H}^0$ and $\mathbf{H}^j$.

Theorem 2: For a t-error-correcting $(n, k, d_{\min})$ *bp* BCH code, if all the decision-vector sets ϕ_ζ $(\zeta = 1, 2, \cdots, t)$ can be found and distinguished from one another, then any error pattern of weight t or less can be corrected by a step-by-step decoding method.

Proof:

Case 1: If the weight of the received error pattern is 1, then $\mathbf{H}^0 \in \phi_1$. Consider temporarily changing the received digits $r_{n-1}, \cdots, r_0$ one at a time. Suppose that r_{n-j} is an erroneous bit; then changing r_{n-j} will reduce the weight-of-error pattern and hence $\mathbf{H}^j \in \phi_0$. Conversely, suppose r_{n-j} is a correct bit; then changing r_{n-j} will increase the weight-of-error pattern to two and hence $\mathbf{H}^j \in \phi_2$. Since ϕ_0, ϕ_1, and ϕ_2 can be distinguished from one another, the error pattern can be correctly decoded.

Case v, $2 \le v < t$: If the weight of the received error pattern is v, then $\mathbf{H}^0 \in \phi_v$. Consider temporarily changing the received digits $r_{n-1}, \cdots, r_{v-1}$ one at a time. Suppose r_{n-j} is an erroneous bit; then changing r_{n-j} will make $\mathbf{H}^j \in \phi_{v-1}$. Suppose r_{n-j} is a correct bit; then changing r_{n-j} will make $\mathbf{H}^j \in \phi_{v+1}$. Since ϕ_{v-1}, ϕ_v, and ϕ_{v+1} can be distinguished from one another, the error can be corrected. After the first error has been corrected, this case is reduced to the case $(v-1)$.

Case t: If the weight of the received error pattern is t, then $\mathbf{H}^0 \in \phi_t$. Consider temporarily changing the received symbols $r_{n-1}, \cdots, r_{t-1}$ one at a time. Suppose r_{n-j} is an erroneous bit; then changing r_{n-j} will make $\mathbf{H}^j \in \phi_{t-1}$. Suppose r_{n-j} is a correct bit; then changing r_{n-j} will increase the weight-of-error pattern to $t+1$. That is, the weight of $e(x) + x^j$ is $t+1$. For a t-error-correcting *bp* BCH code, $d_{\min} \ge 2t+1$, and thus it is possible for some received words to have $r(x) + x^{n-j} = C(x) + e(x) + x^{n-j} = C'(x) + e'(x)$ where $C'(x)$ is another code word and the weight of $e'(x)$ is at least t. Clearly, Hamming distance of $\{C, C'\}$ = weight of $\{e + e'\}$. Therefore, the decision vectors of $e'(x)$ and $e(x) + x^{n-j}$ can be discriminated with any other decision vector belonging to the decision-vector set ϕ_{t-1}. Besides, ϕ_t can be distinguished from ϕ_{t-1}. Therefore, the error can be corrected. After the first error has been corrected, this case is reduced to the case $(t-1)$.

In summary, any combination of t or less errors can be decoded correctly with a step-by-step method.

Based on Theorem 2, a modified step-by-step decoding algorithm for decoding a t-error-correcting *bp* BCH code can be described as follows:

0) Let $j = 0$.
1) Calculate syndrome values S_i^0 $(i = 1,3,\cdots,2t-1)$ from $r(x)$.
2) Obtain $\mathbf{H}^0$.
3) Let $j = j + 1$.
4) Shift syndrome values once; calculate $S_i^j(x) + 1$ $(i = 1,3,\cdots,2t-1)$ and then obtain $\mathbf{H}^j$.
5) Let $\mathbf{H}^0 \in \phi_\zeta$ and $\mathbf{H}^j \in \phi_{\zeta-1}$ (where $1 \le \zeta \le t$), then perform: $r_{n-j} = r_{n-j} + 1$.
6) If $j = n$, then pass the check bit r_0 without decoding; otherwise, go to step 3.

This modified step-by-step decoding algorithm needs $2n$ operation cycles to decode one received word. In the first n operation cycles, initial syndrome values S_i^0 are calculated in step 1. In the $n + 1$th operation cycle, $\mathbf{H}^0$ is calculated. In the other $n - 1$ operation cycles, the errors in the received word, except r_0, are corrected. Since we only concern the errors in the information part of the code, the decoding work of check bit r_0 can be skipped without effecting the performance. The reason of skipping r_0 without decoding is to achieve a real-time decoding in hardware implementation. The number of total operation cycles for finding $\mathbf{H}^j$ $(j = 0,1,2,\cdots,n-1)$ is just equal to n, which is equal to the requirement number of operation cycles for calculating initial syndrome values. Therefore, when the decoder is used for finding $\mathbf{H}^j$ of the current received word, the initial syndrome values of the next received word can be concurrently calculated by another extra syndrome generator [3]. Thus, the average operation cycle for decoding each received word is equal to the block length of the code. The detailed operation procedure of the real-time decoder will be described in the next section.

IV. Hardware Implementation

A high-speed real-time decoder based on the above decoding algorithm is proposed in the following. Fig. 1 shows the functional block diagram of the decoder. The decoder comprises one syndrome generator, one shifted syndrome generator, one comparison module, and one error corrector. The first syndrome generator is used to calculate the initial syndrome values of received words, S_i^0 $(i = 1,3,\cdots,2t-1)$, that is, step 1 of the modified step-by-step decoding algorithm. The second shifted syndrome generator is used to obtain shifted-syndrome values, $S_i^1, S_i^2,\cdots,S_i^{n-1}$ in sequence. Both the syndrome generator and the shifted syndrome generator can be implemented by conventional LFSR's [1]–[5], or by systolic circuits. The comparison module is used to calculate the temporarily changed syndrome values $\overline{S}_i^j$ $(0 \le j \le n-1;\ i = 1,3,\cdots,2t-1)$ and then determine the decision bits h_p^j $(0 \le j \le n-1;\ p = 1,2,\cdots,t)$. According to the decision bits, h_p^0 $(p = 1,2,\cdots,t)$ and h_p^j $(1 \le j \le n-1;\ p = 1,2,\cdots,t)$, the error corrector can tell whether the first position of $r^j(x)$, r_{n-j}, is erroneous or not. If the corresponding bit is judged to be an erroneous bit, the decoder sends a correcting bit $E_c = 1$ to change its magnitude. The detailed design of each module of the decoder is described as follows.

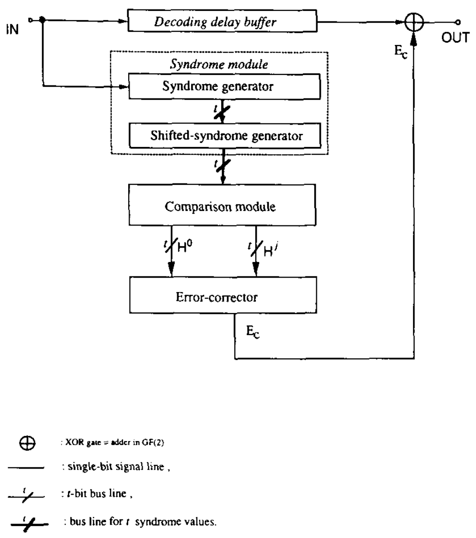


Fig. 1. Functional block diagram of the high-speed real-time t-error-correcting *bp* BCH decoder.

A. LFSR Syndrome Generator

It is well known that a conventional syndrome generator can be implemented by t pieces of LFSR's [1]–[5]. A combined design of syndrome generator and shifted syndrome generator is proposed by slightly modifying the conventional syndrome generator, as shown in Fig. 2. The circuit showed in Fig. 2 is an example of (15, 7, 5) *bp* BCH code, and the architecture can be analogously extended for any other codes. As soon as the entire $r(x)$ has been shifted into the upper LFSR's, the contents in the upper LFSR's are saved in mt pieces of latches that will be used to reset the contents of the lower LFSR's at the $(n + 2)$th clock cycle by a control sequence CS1 $= (1, 0^{n-1})_n^1$, where symbol $(1, 0^{n-1})_n^d$ denotes a periodic bit sequence having a period of n bits and having d delay bits preceding the first bit of the bit sequence. Each of the delay bits is with a "don't-care" value, as shown in Fig. 3. After the setting, S_i^0 can be obtained at the output. Continuously shifting the lower LFSR $n - 1$ times, we can find $S_i^1, S_i^2,\cdots,S_i^{n-1}$ at the output in sequence. In S_3^j circuit of Fig. 2, some latches are inserted between the lower LFSR and the adders to speed up the computation of the syndrome module. The design rule is to make the computation time less than two logic-gate delays, which is the average computation time of the comparison module. In S_1^j circuit of Fig. 2, the same number of latches are also inserted to let all the syndrome values arrive the comparison module at the same time.

The latency of the syndrome module depends on the insertion of latches, where the latency is defined as the group delay of first-input and first-output bit. For example, the latency of Fig. 2 is $n + 4$.

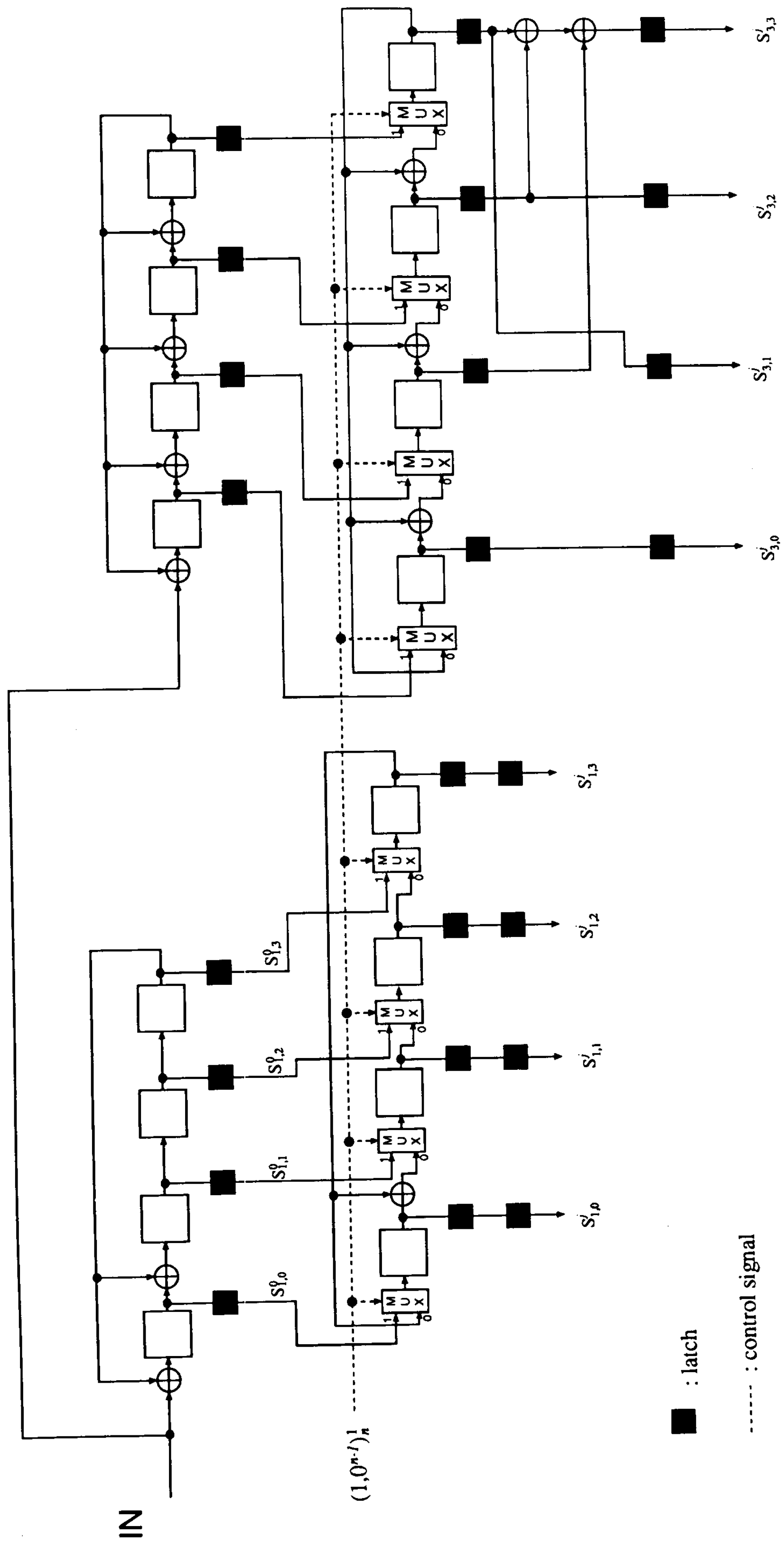


Fig. 2. Syndrome module of (15, 7, 5) *bp* BCH decoder.

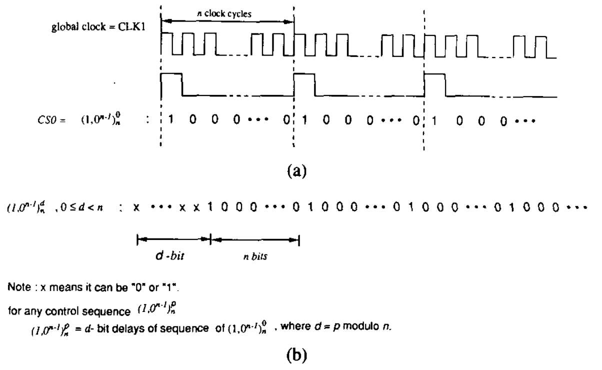

global clock = CLK1
n clock cycles
CS0 = $(1,0^{n-1})_n^0$
1 0 0 0 ··· 0 1 0 0 0 ··· 0 1 0 0 0 ···
(a)
$(1,0^{n-1})_n^d$, $0 \le d < n$: x ··· x x 1 0 0 0 ··· 0 1 0 0 0 ··· 0 1 0 0 0 ··· 0 1 0 0 0 ···
d-bit
n bits
Note : x means it can be "0" or "1".
for any control sequence $(1,0^{n-1})_n^p$
$(1,0^{n-1})_n^p$ = d- bit delays of sequence of $(1,0^{n-1})_n^0$, where d = p modulo n.
(b)

Fig. 3. Required control sequences of the real-time *bp* BCH decoder. (a) Global clock and basic control sequence. (b) Required control sequences.

B. Systolic Syndrome Generator

Since a feedback connection is required in an LFSR circuit, the speed of shift operation will be affected because of propagation delay of long feedback wire. In general, the degradation in speed depends on the layout of feedback-connection wire and its length. For small m, the operation of LFSR can still be very fast. When the decoding speed is very critical, based on (5), a systolic syndrome generator is presented [11]. The syndrome generator consists of t cells, where the cell circuit is redrawn in Fig. 4. When the syndrome values are calculated in the syndrome generator, the values will be kept there by a control sequence $\text{CS1} = (1,0^{n-1})_n^0$ until the syndrome values of the next received word are calculated. Clearly, the latency of the systolic syndrome generator is n clock cycles.

The shifted syndrome generator can also be implemented by systolic circuits. Since S_i^j can be further expressed by

$$\begin{aligned} S_i^j(\alpha) &= r^j(x)_{|x=\alpha^i} \\ &= \operatorname{mod}\{r(x)x^j/x^n + 1\}_{|x=\alpha^i} \\ &= S_i^0(\alpha)\alpha^{ji}, \qquad 0 \le j \le n;\, i = 1,3,\cdots,2t-1. \end{aligned} \tag{10}$$

A cell circuit of a systolic shifted syndrome generator based on (10) is shown in Fig. 5. To perform the multiplication operation in GF(2^m), a parallel-in, parallel-out product-sum systolic multiplier can be employed, where the average computation time of a multiplication is only two gate delays [10]. The latency of the systolic multiplier is $2m$ clock cycles. A shifted syndrome generator consists of t identical cells. The shifted syndrome generator keeps the syndrome values in the first cycle, then cyclically shifts the syndrome values once in each multiplication, that is, obtaining $S_i^0, S_i^1, S_i^2, \cdots, S_i^{n-1}$ in sequence. Since some latches are added at the output of the multiplier to let all of the bits of syndrome value arrive the output at the same time, the latency of systolic shifted syndrome generator is $3m - 1$ clock cycles.

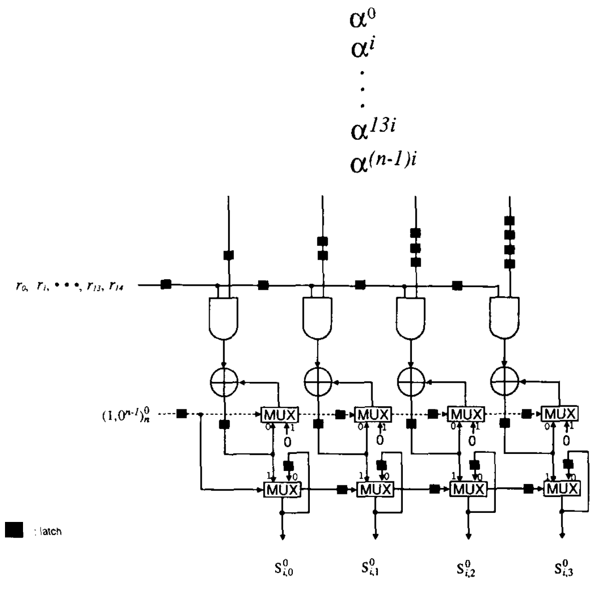


Fig. 4. Cell circuit of systolic syndrome generator over GF(2^4). From [11].

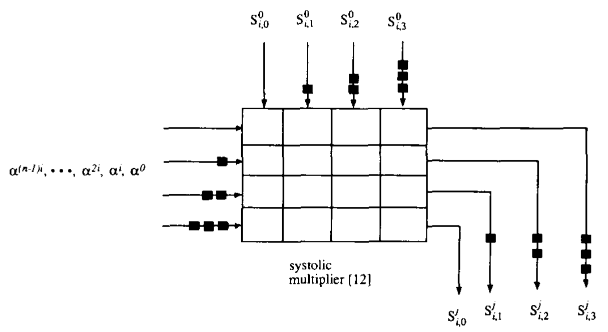


Fig. 5. Cell circuit of systolic shifted syndrome generator.

Hereinafter, all the control sequences used in the decoder are based on the assumption that the systolic syndrome generator and shifted syndrome generator are employed.

C. Comparison Module

Fig. 6 is a block diagram of a comparison module, where t pieces of simple complement circuits are used to pass $S_i^0 = \overline{S}_i^0$, $i = 1,3,\cdots,2t-1$ in parallel at the first clock cycle and then used to obtain $S_i^j + 1 = \overline{S}_i^j$ for $j = 1,2,\cdots,n-1$. The operation is controlled by a control sequence $\text{CS2} = (1,0^{n-1})_n^{3m-1}$. When CS2 = 1, the complement circuits pass the first bit of syndrome values; when CS2 = 0, the complement circuits complement the magnitude of the first bit of syndrome values. Fig. 7 shows the circuit of a complement circuit. To determine the decision bits h_p^j $(0 \le j \le n-1;\, p = 1,2,\cdots,t)$ in terms of $\overline{S}_i^j$ $(0 \le j \le n-1;\, i = 1,3,\cdots,2t-1)$, we must first calculate the values of $\det(\mathbf{L}_p^j)$ $(0 \le j \le n-1;\ p = 1, 2, \cdots, t)$. In Fig. 6, the matrix-calculation circuit, a subcircuit in the comparison module, is used to calculate the determinant of the syndrome matrix, $\det(\mathbf{L}_1^j)$. Only addition and multiplication operations are required for computing the determinant of the syndrome matrix. The addition operation in GF(2^m) is quite simple and can be accomplished by using a set of m pieces of 2-input exclusive-OR (XOR) gates. Since the multiplication operation is performed by a product-sum systolic multiplier with a latency of $2m$ clock cycles [12], the latency of the matrix-calculation circuit is determined by the number of multipliers. To reduce the overall latency of the matrix-calculation circuit, a power-sum systolic circuit can also be employed [13]. The design of matrix-calculation circuits for double- and triple-error-correcting *bp* BCH codes will be illustrated in the later section. Finally, after finding the values of $\det(\mathbf{L}_p^j)$, the decision bits h_p^j $(0 \le j | n-1;\ p = 1, 2, \cdots, t)$ can be determined by using t simple zero-checking circuits, each one constructed by an m-input NOR gate and some latches, as shown in Fig. 8. The t refresh circuits are cascaded with the zero checkers. There are two parallel outputs in the refresh circuit: the right output represents the initial decision bits, h_p^0; while the left output pin represents the h_p^j. The write-in operation of h_p^0 is controlled by the control sequence CS3 $= (1, 0^{n-1})_n^{g_d+4m}$. As the value h_p^0 is saved, it will be kept unchanged in the next $n-1$ clock cycles. Clearly, the values appearing in the two output pins of the refresh circuit will be the same at the write cycle of h_p^0. Finally, it is noted that the calculation of decision bits h_p^j for all j can be obtained by using the same circuits, since it only depends on the circuit input, $\bar{S}_i^j$.

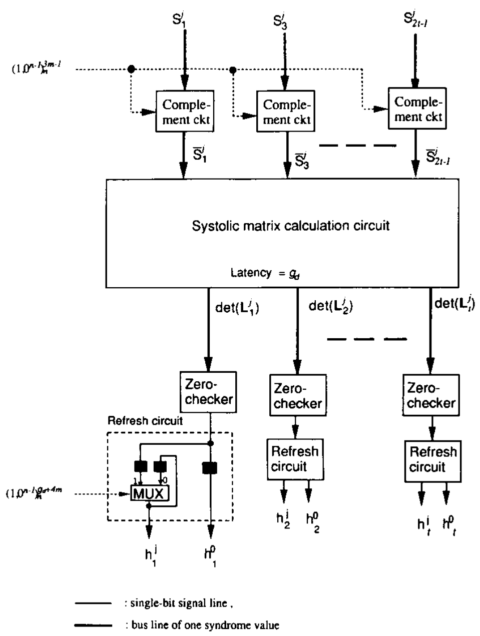


Fig. 6. Comparison module of the real-time decoder for t-error-correcting *bp* BCH codes.

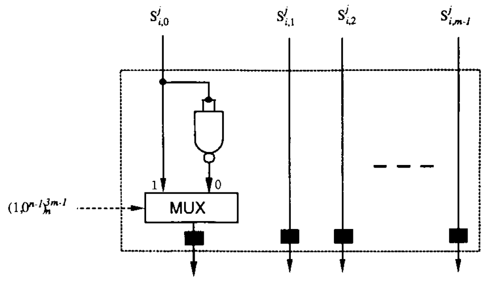


Fig. 7. Circuit of complement circuit.

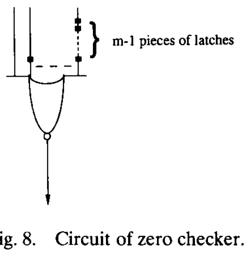


Fig. 8. Circuit of zero checker.

D. Error Corrector

The error corrector is used to perform the operation of step 5 of the modified step-by-step decoding algorithm. When the decision vectors $\mathbf{H}^0$ and $\mathbf{H}^j$ are determined, the error corrector can then determine whether the corresponding bit is erroneous or not in terms of the difference between $\mathbf{H}^0$ and $\mathbf{H}^j$. The error corrector can be easily implemented by some logic gates. After the decision, the circuit sends a correcting bit $E_c = 1$ or $E_c = 0$ to decode the corresponding bit. Some latches may need to be added in the error corrector to make the average computation time equal to or less than two logic-gate delays, which is the computation time in the syndrome module and comparison module. The logic function of the error corrector is determined only by t and is independent of the block length of the code. Based on the logic function of the modified decoding algorithm, it is found that the output of the error corrector, E_c, is always equal to 0 in the writing cycle of $\mathbf{H}^0$ (within this cycle, $\mathbf{H}^j = \mathbf{H}^0$ for any j).

E. Operation and Control Sequences

Fig. 9 shows the operation principle of the high-speed real-time decoder. The received words are consecutively read in. After n clock cycles, the initial syndrome values of the first received word are calculated in the syndrome generator and then passed to the shifted syndrome generator, as shown in Figs. 2 and 4. At the same time, the syndrome generator is ready for calculating the initial syndrome values of next received word, that is, the second received word is consecutively read into the syndrome generator without interrupting. After $g_d + 4m$ clock cycles, the decision vector $\mathbf{H}^0$ is obtained at the output of

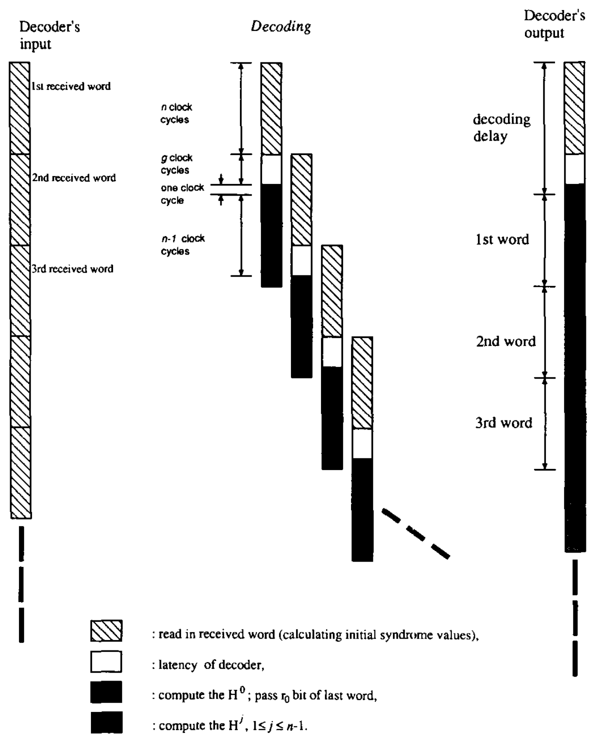


Fig. 9. Operation sequence of the real-time decoder.

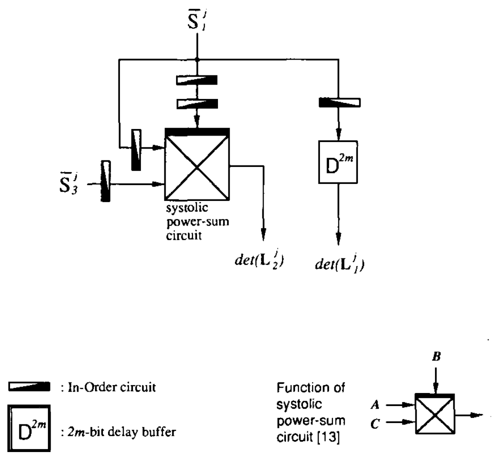

S̄ʲ₁
S̄ʲ₃
systolic power-sum circuit
D²ᵐ
det(Lʲ₂)
det(Lʲ₁)
: In-Order circuit
: 2m-bit delay buffer
Function of systolic power-sum circuit [13]
A
B
C

Fig. 10. Matrix-calculation circuit of $(n, k, 5)$ *bp* BCH decoder.

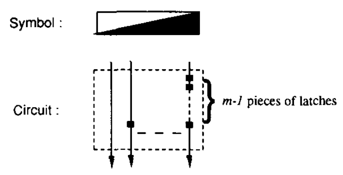


Fig. 11. In-order circuit

the comparison module. When the first $(n - 1 - g_d - 4m)$ bits of the first received word are decoding, i.e., after $2n$ clock cycles of the global clock, the initial syndrome value of the second received word is found in the syndrome generator. After the first $n - 1$ bits of the first received word are decoding, the $\mathbf{H}^0$ of the second received word is consecutively sent to the refresh circuit of the comparison module, that is, the r_0 bit of the first received word is directly read out from the buffer without decoding at the cycle of finding the $\mathbf{H}^0$ of the second received word. Fortunately, bit r_0 is a check bit when the received word is in systematic form. Repeating the same process, the high-speed real-time decoder may work at a speed equal to the line-data rate, with a group delay of $n + g + 1$ clock cycles at the initial time, where g is the latency of the decoder and the extra one clock delay is used for finding $\mathbf{H}^0$ of the first received word.

The required global-control clock signal CLK1 of the real-time decoder is shown in Fig. 3. All the shift operations of latches and registers of the decoder are controlled by the pulse lead of CLK1. In practice, the basic clock signal CLK1 can be extracted from the line signal by employing a phase-locked-loop (PLL) circuit. As shown in Figs.1–8, the high-speed real-time decoder requires only three control sequences to do the decoding work. The first control sequence $\text{CS1} = (1, 0^{n-1})_n^0$ or $\text{CS1} = (1, 0^{n-1})_n^1$ is used to calculate the initial syndrome values and pass them to the shifted syndrome generator. The second control sequence $\text{CS2} = (0, 1^{n-1})_n^{3m-1}$ is used to complement the first bit of syndrome values. The third control sequence $\text{CS3} = (1, 0^{n-1})_n^{g_d+4m}$ is used to save the decision bit h_p^0. All the three control sequences can be generated from the basic control sequence $(1, 0^{n-1})_n^0$ by some delay latches.

V. Design Examples

A. Double-Error-Correcting bp BCH Codes

Fig. 10 shows the circuit diagram of a matrix calculation circuit. It needs only a power-sum circuit, which is composed of m^2 identical cells [13] and some in-order circuits to control the input bit sequences [12], [13]. The in-order circuit can be constructed by some latches, as shown in Fig. 11. The latency of the matrix calculation circuit is only $2m$, and the latency of the comparison circuit is therefore equal to $3m + 1$. Fig. 12 shows the comparison module of a $(15, 7, 5)$ *bp* BCH decoder as an example. In the fresh circuits, since the first 25 bits of $(1, 0^{14})_{15}^{24}$ can be of any pattern, as shown in Fig. 3, the control sequence $(1, 0^{14})_{15}^{24}$ can be substituted by $(1, 0^{14})_{15}^{9}$ without affecting the decoding process. For the double-error-correcting *bp* BCH code, $\phi_0 = \{(1,1)\}$, $\phi_1 = \{(0,1)\}$, and $\phi_2 = \{(0,0)\}$. Based on the decision vectors, Fig. 13 shows the corresponding error corrector. The latency of the error corrector is only one clock cycle. Thus, considering one clock delay of finding $\mathbf{H}^0$, the decoding delay of double-error-correcting *bp* BCH code can be found to be $n + 6m + 2$, which is the required length of the decoding delay-buffer in Fig. 1.

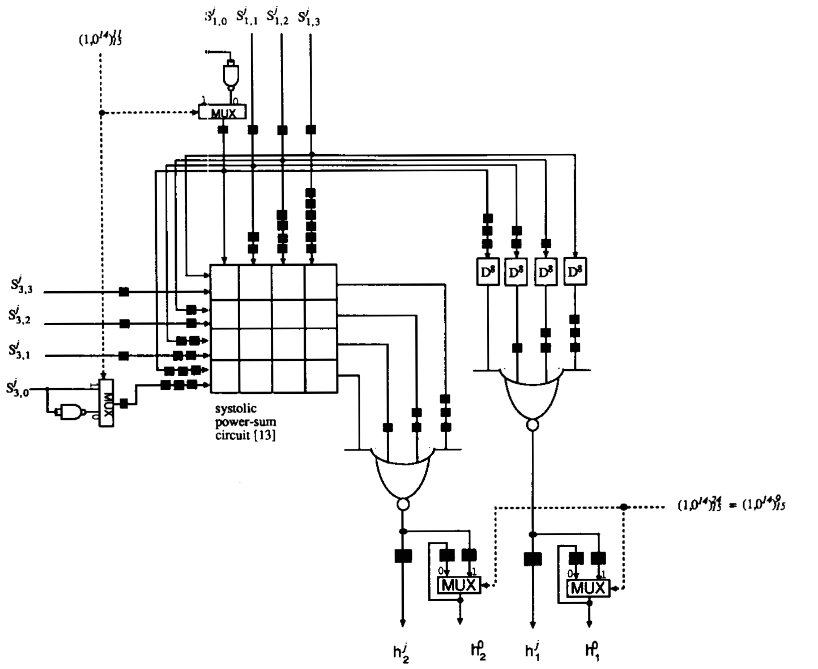


Fig. 12. Comparison module of $(15, 7, 5)$ *bp* BCH decoder.

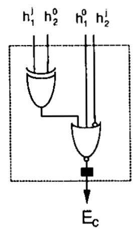


Fig. 13. Error corrector of $(n, k, 5)$ *bp* BCH decoder.

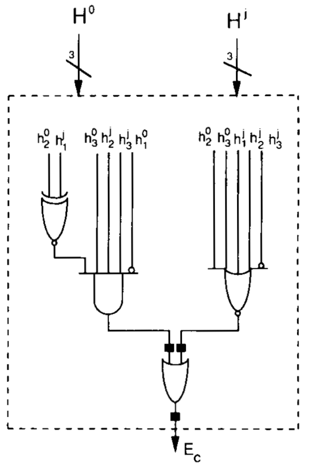

H^0
H^j
3
3
h^0_2 h^j_1 h^0_3 h^j_2 h^j_3 h^0_1
h^0_2 h^0_3 h^j_1 h^j_2 h^j_3
E_c

Fig. 14. Error corrector of $(n, k, 7)$ *bp* BCH decoder.

B. Triple-Error-Correcting bp BCH Codes

From Theorem 1, the decision-vector sets of triple-error-correcting *bp* BCH codes are $\phi_0 = \{(1, 1, 1)\}$, $\phi_1 = \{(0, 1, 1)\}$, $\phi_2 = \{(0, 0, 1)\}$, and $\phi_3 = \{(1, 0, 0), (0, 0, 0)\}$. The error corrector for the $(n, k, 7)$ *bp* BCH codes is implemented in Fig. 14 by using the decision-vector sets. The latency of the error corrector is two clock cycles. From a hardware-implementation point of view, the computation path in the matrix-calculation circuit should be designed carefully; a well-planned organization of computation paths will make the latency of the matrix-calculation circuit small. For example, in this case of triple-error-correcting *bp* BCH code, computations of $\det(\mathbf{L}_1^j)$, $\det(\mathbf{L}_2^j)$, and $\det(\mathbf{L}_3^j)$ in the matrix-calculation circuit are required. The operations of $\det(\mathbf{L}_1^j)$ and $\det(\mathbf{L}_2^j)$ are the same as that in the double-error-correcting case. The expression $\det(\mathbf{L}_3^j) = (\bar{S}_1^j)^6 + (\bar{S}_1^j)^3\bar{S}_3^j + \bar{S}_1^j\bar{S}_5^j + (\bar{S}_3^j)^2$ can be reorganized as $[(\bar{S}_1^j)^3 + \bar{S}_3^j]^2 + [(\bar{S}_1^j)^2\bar{S}_3^j + \bar{S}_5^j]\bar{S}_1^j$, then only one product-sum multiplier, three power-sum circuits, and one adder are required to compute $\det(\mathbf{L}_3^j)$. The detail of the design of a matrix-calculation circuit is illustrated in Fig. 15. It can be seen that the latency of the matrix-calculation circuit is only $5m$ clock cycles. Thus, the decoding delay of the triple-error-correcting *bp* BCH code is $n + 9m + 3$.

VI. Conclusions

A modified step-by-step decoding algorithm for t-error-correcting *bp* BCH codes has been presented. The decoding algorithm avoids the need to calculate the error-location polynomial in order to find the error locators. Also, by avoiding the use of computation-intensive

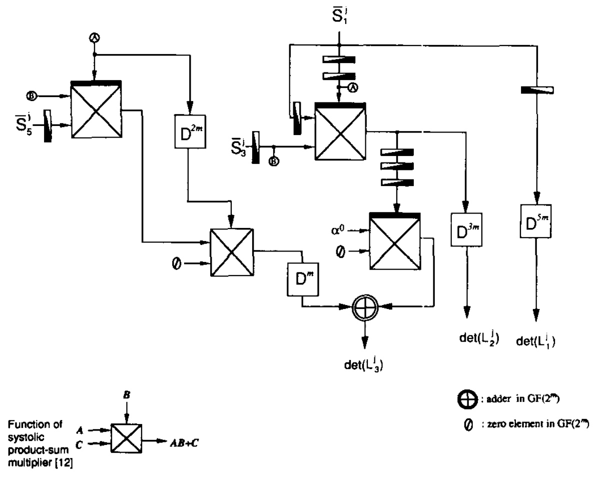


Fig. 15. Matrix calculation circuit of $(n, k, 7)$ *bp* BCH decoder.

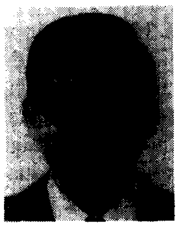
inverse operations, the modified decoding method can be much less complex than the conventional standard algebraic method in hardware implementation. Based on the modified step-by-step decoding algorithm, a high-speed real-time *bp* BCH decoder has been presented. The decoding speed of this decoder can be up to the inverse of two logic-gate delays. Based on different VLSI technologies, the decoder can be operated from several hundred megabits per second up to the order of gigabits per second. The decoder requires only three control sequences, which can be generated by a basic control sequence. The detailed circuits of the matrix-calculation circuit and error-corrector of double- and triple-error-correcting codes are also given. Because of its simplicity in structure and circuit realization, this decoder may be easily implemented in one monolithic chip.

References

[1] S. Lin and D. J., Costello, Jr., *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
[2] A. M. Michelson and A. H. Levesque, *Error-Control Techniques for Digital Communication*. New York: Wiley, 1985.
[3] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. Cambridge, MA: M.I.T. Press, 1972.
[4] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.
[5] C. C. Clark and J. B. Cain, *Error Correcting Coding for Digital Communications*. New York: Plenum, 1981.
[6] W. W. Peterson, "Encoding and error-correcting procedures for the Bose-Chaudhuri codes," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 459-470, Sept. 1960.
[7] J. L. Massey, "Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inform. Theory*, vol. IT-11, no. 4, pp. 580-585, Oct. 1965.
[8] S. W. Wei and C. H. Wei, "High speed hardware decoder for double-error-correcting binary BCH codes," *Inst. Elec. Eng. Proc.*, vol. 136, no. 3, pp. 227-231, June 1989.
[9] M. Kubo, I. Masuda, K. Miyata, and K. Ogiue, "Perspective on BiCMOS VLSI's," *IEEE J. Solid-State Circuits*, vol. 23, no. 1, pp. 5-11, Feb. 1988.
[10] H. Morkoç, "MODFET's: Soar to 400 GHz," *IEEE Circuits Devices Mag.*, vol. 7, no. 6, pp. 14-20, Nov. 1991.
[11] C.-L. Wang and W.-J. Bair, "A VLSI architecture for implementation of the decoder for binary BCH codes," in *Proc. Int. Symp. Commun.*, (Taiwan), Dec. 9-13, 1991, pp. 36-40.
[12] C.-S. Yeh, Irving S. Reed, and T. K. Truong, "Systolic multipliers for finite fields GF(2^m)," *IEEE Trans. Comput.*, vol. C-33, no. 4, pp. 357-360, Apr. 1984.
[13] S. W. Wei, "A systolic power-sum circuit for GF(2^m)," in *Proc. Int. Sympos. Commun.*, (Taiwan), Dec. 9-13, 1991, pp. 61-64.

Shyue-Win Wei (S'85-M'86-S'88-M'90) was born in Taiwan on June 9, 1958. He received the B.S. degree in telecommunications from Central Police College, Taiwan, R.O.C., in 1980, the M.S. degree in communications and the Ph.D. degree in electronics from National Chiao Tung University, Hsinchu, Taiwan, in 1986 and 1990, respectively.

From 1980 to 1984 he worked at the Institute of Police Telecommunications, Taiwan. In 1990 he joined Telecommunications Laboratories, Chung-Li, Taiwan, where he worked on the development of a high-bit-rate digital subscriber-line transmission system. Since 1992 he has been Associate Professor in the Department of Electrical Engineering, Chung Hua Polytechnic Institute. His research interests include digital transmission system, digital subscriber lines, coding theory, and VLSI, implementation.



Che-Ho Wei (S'73-M'76-M'79-SM'87) was born in Taiwan in 1946. He received the B.S. and M.S. degrees in electronic engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, R.O.C., in 1968 and 1970, respectively, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1976.

From 1976 to 1979, he was an Associate Professor at NCTU, where he is now a Professor in the Department of Electronics Engineering and the Institute of Electronics. From 1979 to 1982, he was the Engineering Manager of Wang Industrial Company in Taipei, Taiwan. He was the Chairman of the Department of Electronics Engineering of NCTU from 1982 to 1986 and Director of the Institute of Electronics from 1984 to 1989. He served as Associate Director of the Microelectronics and Information Science and Technology Research Center of NCTU from February to August 1990. He was on leave from the Ministry of Education and served as Director of the Advisory Office from September 1990 to July 1992.

Dr. Wei was the founding chairman of both the IEEE Circuits and Systems Society and IEEE Communication's Society chapters in Taipei. He received the Outstanding Research Award in 1987-1989 and the Distinguished Research Award in 1990 from the National Science Council, Taiwan, R.O.C. His research interests include digital communications, signal processing, and related VLSI circuits design.