Computers Math. Applic. Vol. 26, No. 7, pp. 59–66, 1993
Printed in Great Britain.

0898-1221/93 $6.00 + 0.00

An Improved Algorithm for Davida and Cowles's Decoding Method

T. HWANG
Institute of Information Engineering
National Cheng Kung University, Tainan, Taiwan R.O.C.

S. Y. HWANG
Institute of Applied Mathematics
National Cheng Kung University, Tainan, Taiwan R.O.C.

(Received May 1992; revised and accepted September 1992)

Abstract—In this paper, we apply Chio's pivotal condensation process for matrix determinant evaluation to speed up the decoding algorithm of Davida *et al.* for binary BCH codes. A comparison is given to demonstrate the merit of the modified scheme.

Keywords—BCH codes, Chio's pivotal condensation process, Error-correcting codes, Laplace expansion method, Step-by-step decoding.

1. INTRODUCTION

The Bose-Chaudhuri-Hocquenghem (BCH) codes are a class of widely studied error-correcting codes. Considerable work has been done on the decoding of these codes. They can be described briefly as follows.

Let α be an element in GF(q^m), an extension field of the Galois field GF(q). For any given positive integers m_0 and d, if $g(x)$ is the lowest degree polynomial over GF(q) that has α^{m_0}, $\alpha^{m_0+1}, \ldots, \alpha^{m_0+d-2}$ as roots, then the code generated by $g(x)$ is a (q, m_0, d) BCH code. The length of the code can be computed from the least common multiple of the orders of the roots. As a special case, if α is primitive element over GF(2^m) and $m_0 = 1$, then $g(x)$ generates the primitive binary BCH codes [1,2].

In this paper, we consider the primitive t-error-correcting binary code of length n that has α, $\alpha^2, \ldots, \alpha^{2t}$ as its roots. Suppose that code vector $v(x) = v_0 + v_1 x + v_2 x^2 + \cdots + v_{n-1} x^{n-1}$ is transmitted and $r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1} x^{n-1}$ is received. Let $e(x)$ be the error pattern, i.e., $r(x) = e(x) + v(x)$. If we assume that e ($e \leq t$) errors occured in $r(x)$, then the syndrome, $\mathbf{S} = (s_1, s_2, \ldots, s_{2t})$ of $r(x)$ can be defined as

$$s_j = r(\alpha^j) = e(\alpha^j), \qquad 1 \leq j \leq 2t.$$

Let

$$\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x) \ldots (1 + \beta_e x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_e x^e,$$

where $\beta_i = \alpha^{l_i}$, the coefficient of the term of degree l_i in the code polynomial for $i = 1, 2, \ldots, e$, is the element of the error location number. The roots of $\sigma(x)$ are $\beta_1^{-1}, \beta_2^{-1}, \ldots, \beta_e^{-1}$, which are

The authors would like to thank the referees of this paper for their helpful comments and suggestions. This paper was supported by the NSC of ROC under the contract number NSC 80-0408-E-006-07.

the inverses of the error location numbers. The polynomial $\sigma(x)$ is used conventionally to locate the error positions and thus is called the error-locating polynomial [3].

In 1975, Davida and Cowles proposed a new error-locating polynomial to decode binary BCH codes [4]. Suppose e is the number of errors in the received vector. They defined a new error-locating polynomial as

$$\hat{\sigma}_e(x) = \prod_{i=1}^{e} (x + \beta_i) \prod_{\substack{1 \le k, l \le e \\ l > k}}^{e} (\beta_l + \beta_k),$$

which can be determined by the matrix

$$\mathbf{M}_e = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ s_2 & s_1 & 1 & \dots & 0 \\ s_4 & s_3 & s_2 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ s_{2e-2} & s_{2e-3} & s_{2e-4} & \dots & s_{e-1} \end{pmatrix}. \tag{1}$$

THEOREM 1. *[5] For any BCH $(2, 1, d)$ code and any j such that $2 \le j \le n$, the $j \times j$ matrix*

$$\mathbf{M}_j = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ s_2 & s_1 & 1 & \dots & 0 \\ s_4 & s_3 & s_2 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ s_{2j-2} & s_{2j-3} & s_{2j-4} & \dots & s_{j-1} \end{pmatrix}$$

is singular if the weight of $e(x)$ is $j - 2$ or less, and is non-singular if the weight of $e(x)$ is $j - 1$ or j.

THEOREM 2. *[4] For an binary BCH code, let $\mathbf{M}_e(s_1, s_2, \dots, s_{2e-2})$ be the matrix in (1); then $\hat{\sigma}_{e-1}(x) = |\mathbf{M}_e(x + s'_1, x^2 + s'_2, \dots, x^{2e-2} + s'_{2e-2}|$ where $s'_i = s_i + \beta_e^i$.*

Let t be the error-correcting capability of the introduced BCH code; Davida *et al.* developed a decoding scheme based on this error-correcting polynomial as follows.

1.1. Davida *et al.*'s Decoding Algorithm

Begin

Step 1. Set $e = t$.

Step 2. Compute $|\mathbf{M}_e|$.

Step 3. If $|\mathbf{M}_e| = 0$, then set $e = e - 2$ and go to Step 2. Otherwise, go to Step 4.

Step 4. $\hat{\sigma}_e(x) = |\mathbf{M}_{e+1}(x + s'_1, x^2 + s'_2, \dots, x^{2e} + s'_{2e})|$.

Step 5. Find the roots of $\hat{\sigma}_e(x)$ by Chien search.

End.

The decoding scheme is similar to Massey's step-by-step decoding algorithm [5]. In order to determine the number of errors, both schemes test whether the matrix $\mathbf{M}_t$ is singular or not. If not, they successively test $\mathbf{M}_{t-2}$, $\mathbf{M}_{t-4}, \dots$, until the first nonsingular matrix $\mathbf{M}_e$ $(1 \le e \le t)$ appears. Once the number of errors is obtained, one has to evaluate the determinant of a matrix of rank $e+1$ at Step 4, and finally use Chien search [1] to locate the errors. Hence, the computation of the determinant of a matrix is the most important work in Davida *et al.*'s decoding method. In this paper, we shall present a method to speed up this process and also propose a new decoding process such that the error correction can be performed in a simpler way.

2. IMPROVED DECODING ALGORITHM OF DAVIDA *ET AL.*

2.1. Chio's Pivotal Condensation Process

Let $\mathbf{A}_{(i)(j)}$ denote the $(m-1)\times(m-1)$ submatrix of the $m\times m$ matrix $\mathbf{A}$ obtained by deleting the i^{th} row and the j^{th} column of $\mathbf{A}$. First, we study *Chio's Pivotal Condensation Process.*

THEOREM 3. *[6] (Chio's Pivotal Condensation Process) Let* $\mathbf{A}=[a_{ij}]$ *be an* $m\times m$ *matrix and suppose* $a_{11}\neq 0$. *Let* $\mathbf{B}$ *denote the matrix obtained by replacing each element* a_{ij} *in* $\mathbf{A}_{(1)(1)}$ *by* $\begin{vmatrix} a_{11} & a_{1j} \\ a_{i1} & a_{ij} \end{vmatrix}$. *Then* $|\mathbf{A}| = |\mathbf{B}|/a_{11}^{m-2}$. *That is,*

$$|\mathbf{A}| = \frac{1}{a_{11}^{m-2}} \begin{pmatrix} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & \cdots & \begin{vmatrix} a_{11} & a_{1m} \\ a_{21} & a_{2m} \end{vmatrix} \\ \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & \cdots & \begin{vmatrix} a_{11} & a_{1m} \\ a_{31} & a_{3m} \end{vmatrix} \\ \vdots & \vdots & & \vdots \\ \begin{vmatrix} a_{11} & a_{12} \\ a_{m1} & a_{m2} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{m1} & a_{m3} \end{vmatrix} & \cdots & \begin{vmatrix} a_{11} & a_{1m} \\ a_{m1} & a_{mm} \end{vmatrix} \end{pmatrix}.$$

For convenience, we define $\mathbf{A}^{(0)}=\mathbf{A}$ and $\mathbf{B}^{(0)}=\mathbf{B}$ and denote $\mathbf{A}^{(1)}$ as the $(m-1)\times(m-1)$ matrix which is computed by permuting the first row with a nonzero leading element in $\mathbf{B}^{(0)}$ with the first row in $\mathbf{B}^{(0)}$, if the leading element in it is zero. $\mathbf{B}^{(1)}$ denotes the matrix obtained by replacing each element a'_{ij} in $\mathbf{A}^{(1)}_{(1)(1)}$ by $\begin{vmatrix} a'_{11} & a'_{1j} \\ a'_{i1} & a'_{ij} \end{vmatrix}$. Similarly, we define $\mathbf{A}^{(s)}$ and $\mathbf{B}^{(s)}$ for $1\leq s\leq m-2$.

REMARK. If the first column in $\mathbf{B}^{(i)}$, for $1\leq i\leq n$, is a zero-column, then $|\mathbf{A}^{(j)}|=0$, for $i\leq j\leq n$.

COROLLARY 4. *Define* $a_{11}^{(i)}$ *as the element of the first column and the first row in* $\mathbf{A}^{(i)}$. *The determinant of* $\mathbf{A}$ *is*

$$\prod_{i=0}^{m-1}\left(a_{11}^{(i)}\right)^{i+2-m}.$$

PROOF. Since $|\mathbf{A}| = |\mathbf{B}|/a_{11}^{(m-2)}$, therefore, by induction, the determinant value of $|\mathbf{A}|$ is $\left(a_{11}^{(0)}\right)^{2-m}\times\left(a_{11}^{(1)}\right)^{3-m}\times\cdots\times\left(a_{11}^{(m-1)}\right)^{1}=\prod_{i=0}^{m-1}\left(a_{11}^{(i)}\right)^{i+2-m}$. ∎

THEOREM 5. *Let the matrix* $\mathbf{M}_{t+1}$ *be*

$$\mathbf{M}_{t+1}=\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ s_2 & s_1 & 1 & \cdots & 0 \\ s_4 & s_3 & s_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ s_{2t} & s_{2t-1} & s_{2t-2} & \cdots & s_t \end{pmatrix}.$$

By Chio's pivotal condensation process, the matrices $\mathbf{M}_{t+1}^{(e)}$ *and* $\mathbf{M}_{t+1}^{(e+1)}$ *are nonsingular, and* $\mathbf{M}_{t+1}^{(k)}$, *for all* $e+1<k\leq t+1$, *is singular if the weight of the error pattern is* e.

PROOF. According to the Theorem 1, if e or $e-1$ errors occurred, then $|\mathbf{M}_e| \neq 0$. Suppose that e errors occurred, we know that $|\mathbf{M}_{t+1}| = 0$, $|\mathbf{M}_t| = 0$, $|\mathbf{M}_{t-1}| = 0$, $|\mathbf{M}_{t-2}| = 0, \ldots, |\mathbf{M}_{e+1}| \neq 0$ and $|\mathbf{M}_e| \neq 0$. By Chio's pivotal condensation process, we now show that to test whether $\mathbf{M}_e$ is singular or not is the same as to test whether $|\mathbf{M}_{t+1}^{(e)}|$ is singular or not.

Let

$$|\mathbf{M}_{t+1}^{(0)}| = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ s_2 & s_1 & 1 & \cdots & 0 \\ s_4 & s_3 & s_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ s_{2t} & s_{2t-1} & s_{2t-2} & \cdots & s_t \end{pmatrix}$$

$$= \left(\begin{array}{cccc|ccc} m_{11}^{(0)} & m_{12}^{(0)} & \cdots & m_{1e}^{(0)} & m_{1(e+1)}^{(0)} & \cdots & m_{1(t+1)}^{(0)} \\ m_{21}^{(0)} & m_{22}^{(0)} & \cdots & m_{2e}^{(0)} & m_{2(e+1)}^{(0)} & \cdots & m_{2(t+1)}^{(0)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ m_{e1}^{(0)} & m_{e2}^{(0)} & \cdots & m_{ee}^{(0)} & m_{e(e+1)}^{(0)} & \cdots & m_{e(t+1)}^{(0)} \\ \hline \vdots & \vdots & & \vdots & \vdots & & \vdots \\ m_{(t+1)1}^{(0)} & m_{(t+1)2}^{(0)} & \cdots & m_{(t+1)e}^{(0)} & m_{(t+1)(e+1)}^{(0)} & \cdots & m_{(t+1)(t+1)}^{(0)} \end{array}\right)$$

$$= \frac{1}{\left(m_{11}^{(0)}\right)^{t-1}} \left(\begin{array}{ccc|ccc} m_{11}^{(1)} & \cdots & m_{1(e-1)}^{(1)} & m_{1e}^{(1)} & \cdots & m_{1t}^{(1)} \\ m_{21}^{(1)} & \cdots & m_{2(e-1)}^{(1)} & m_{2e}^{(1)} & \cdots & m_{2t}^{(1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ m_{(e-1)1}^{(1)} & \cdots & m_{(e-1)(e-1)}^{(1)} & m_{(e-1)e}^{(1)} & \cdots & m_{(e-1)t}^{(1)} \\ \hline \vdots & & \vdots & \vdots & & \vdots \\ m_{t1}^{(1)} & \cdots & m_{t(e-1)}^{(1)} & m_{te}^{(1)} & \cdots & m_{tt}^{(1)} \end{array}\right)$$

$$= \frac{1}{\left(m_{11}^{(0)}\right)^{t-1}\left(m_{11}^{(1)}\right)^{t-2}} \times \left(\begin{array}{ccc|ccc} m_{11}^{(2)} & \cdots & m_{1(e-2)}^{(2)} & m_{1(e-1)}^{(2)} & \cdots & m_{1(t-1)}^{(2)} \\ m_{21}^{(2)} & \cdots & m_{2(e-2)}^{(2)} & m_{2(e-1)}^{(2)} & \cdots & m_{2(t-1)}^{(2)} \\ \vdots & & \vdots & \vdots & & \vdots \\ m_{(e-2)1}^{(2)} & \cdots & m_{(e-2)(e-2)}^{(2)} & m_{(e-2)(e-1)}^{(2)} & \cdots & m_{(e-2)(t-1)}^{(2)} \\ \hline \vdots & & \vdots & \vdots & & \vdots \\ m_{(t-1)1}^{(2)} & \cdots & m_{(t-1)(e-2)}^{(2)} & m_{(t-1)(e-1)}^{(2)} & \cdots & m_{(t-1)(t-1)}^{(2)} \end{array}\right)$$

$$\vdots$$

$$= \frac{1}{\left(m_{11}^{(0)}\right)^{t-1}\left(m_{11}^{(1)}\right)^{t-2}\cdots\left(m_{11}^{(e-2)}\right)^{t-e+1}}$$

$$\times \begin{pmatrix} m_{11}^{(e-1)} & \vdots & m_{12}^{(e-1)} & \cdots & m_{1\,(t-e+2)}^{(e-1)} \\ \hline m_{21}^{(e-1)} & & m_{22}^{(e-1)} & \cdots & m_{2\,(t-e+2)}^{(e-1)} \\ \vdots & & \vdots & & \vdots \\ m_{(t-e+2)\,1}^{(e-1)} & & m_{(t-e+2)\,2}^{(e-1)} & \cdots & m_{(t-e+2)\,(t-e+2)}^{(e-1)} \end{pmatrix}$$

$$= \frac{1}{\left(m_{11}^{(0)}\right)^{t-1}\left(m_{11}^{(1)}\right)^{t-2}\cdots\left(m_{11}^{(e-1)}\right)^{t-e}} \times \begin{pmatrix} m_{11}^{(e)} & m_{12}^{(e)} & \cdots & m_{1\,(t-e+1)}^{(e)} \\ m_{21}^{(e)} & m_{22}^{(e)} & \cdots & m_{2\,(t-e+1)}^{(e)} \\ m_{31}^{(e)} & m_{32}^{(e)} & \cdots & m_{3\,(t-e+1)}^{(e)} \\ \vdots & \vdots & & \vdots \\ m_{(t-e+1)\,1}^{(e)} & m_{(t-e+1)\,2}^{(e)} & \cdots & m_{(t-e+1)\,(t-e+1)}^{(e)} \end{pmatrix}$$

$$= |\mathbf{M}_e| \frac{1}{\left(m_{11}^{(0)}\right)^{t-e+1}\left(m_{11}^{(1)}\right)^{t-e}\cdots\left(m_{11}^{(e-1)}\right)^{t-2e+2}} \times \begin{pmatrix} m_{11}^{(e)} & m_{12}^{(e)} & \cdots & m_{1\,(t-e+1)}^{(e)} \\ m_{21}^{(e)} & m_{22}^{(e)} & \cdots & m_{2\,(t-e+1)}^{(e)} \\ m_{31}^{(e)} & m_{32}^{(e)} & \cdots & m_{3\,(t-e+1)}^{(e)} \\ \vdots & \vdots & & \vdots \\ m_{(t-e+1)\,1}^{(e)} & m_{(t-e+1)\,2}^{(e)} & \cdots & m_{(t-e+1)\,(t-e+1)}^{(e)} \end{pmatrix}.$$

So, at the process of computing the determinant of the matrix $\mathbf{M}_{t+1}$, we can get the determinant of the matrix $\mathbf{M}_e$ $(e < t+1)$. Since $|\mathbf{M}_e(x)| \neq 0$, if the element of the first row and the first column of the matrix $\mathbf{M}_{t+1}^{(k)}$ $(k < e)$ is zero, then one can always find a row with a nonzero leading element in the matrix $\mathbf{M}_e^{(k)}$ and exchange the first row with this row. Otherwise, $|\mathbf{M}_e^{(k)}| = 0$, which implies $|\mathbf{M}_e^{(l)}| = 0$ if $k \leq l \leq e$. That is, $|\mathbf{M}_e| = 0$, which is a contradiction. Similarly, we can prove that $|\mathbf{M}_{t+1}^{(e+1)}| \neq 0$, $|\mathbf{M}_{t+1}^{(e+2)}| = 0, \ldots, |\mathbf{M}_{t+1}^{(t+1)}| = 0$ if $|\mathbf{M}_{e+1}| \neq 0$ and $|\mathbf{M}_{e+2}| = \cdots = |\mathbf{M}_{t+1}| = 0$. ∎

2.2. Improved Davida *et al.*'s Decoding Algorithm

Theorem 5 facilitates us to calculate the number of errors in $r(x)$ by evaluating the determinant of only one matrix $\mathbf{M}_t$, instead of computing the determinants of $t/2$ matrices as being done in Davida *et al.*'s decoding algorithm.

We further define the *error-number decision vector* $\mathbf{D} = (d_1, d_2, \ldots, d_{t+1})$

$$d_s = \begin{cases} 0, & \text{if } \left|\mathbf{M}_{t+1}^{(s)}(0)\right| = 0 \\ 1, & \text{otherwise} \end{cases} \qquad \text{where } s = 1, 2, \ldots, t, t+1.$$

Then, we have the following:

$\mathbf{D} = (0,0,0,0,\ldots,0,0)$ if there is no error;

$\mathbf{D} = (1,1,0,0,\ldots,0,0)$ if there is one error;

$\mathbf{D} = (1,1,1,0,\ldots,0,0)$ if there are two errors;

$\vdots$

$\mathbf{D} = (1,1,1,1,\ldots,1,0)$ if there are $t-1$ errors;

$\mathbf{D} = (1,1,1,1,\ldots,1,1)$ if there are t errors.

Therefore, by computing the determinant $|\mathbf{M}_{t+1}|$, the number of errors can be obtained. Note, the same method can also be used to improve Massey's step-by-step decoding scheme [5].

Furthermore, at Step 4 of Davida *et al.*'s decoding algorithm, they computed the determinant of the matrix $\mathbf{M}_{e+1}$ first and then found the error positions by Chien's search. However, if we apply Chio's pivotal condensation process directly to locate the error positions, then the computation of the determinant of the matrix $\mathbf{M}_{e+1}$ can be omitted. What we have to do is to substitute all possible error positions, x_i ($1 \leq i \leq n, n$ is the code length), to x subsequently, and test whether $\mathbf{M}_{e+1}(x_i)$ is singular or not. If $|\mathbf{M}_{e+1}(x_i)| = 0$, then the position x_i is in error. In this case, Davida *et al.*'s decoding algorithm can be modified as the following.

Improved Algorithm

Begin

Step 1. Use Chio's pivotal condensation process to evaluate $|\mathbf{M}_{t+1}|$ and also generate $\mathbf{D}$. If $\mathbf{D} = (0, 0, \ldots, 0)$, then end this algorithm; else let the number of errors be e.

Step 2. Calculate $|\mathbf{M}_{e+1}(x_i)|$, for $1 \leq i \leq n$.
If $|\mathbf{M}_{e+1}(x_i)| = 0$, then the position x_i is in error.

End.

It is obvious that Step 2 of the new decoding algorithm involves a singularity test of n matrices. However, the intentional design of the decoding algorithm (see Section 3) to consecutively utilize the singularity test of a matrix in each decoding step allows us to perform these decoding steps in a pipelined way. This will speed up the decoding speed and also simplify the design of hardware circuits.

3. HARDWARE DESIGN AND CONCLUSIONS

We use Chio's pivotal condensation process to evaluate the determinant of a $t \times t$ matrix $\mathbf{A}$. From Theorem 2, we see that the $t \times t$ matrix is first reduced to a $(t-1) \times (t-1)$ matrix and then a $(t-2) \times (t-2)$ matrix, etc. In each stage, it involves the determinant calculations of several 2×2 matrices, each of which can be implemented by using two multipliers and one subtractor (or adder in GF(2)), (see also Figure 1). Figure 2 shows an interconnection diagram of two stages—a 4×4 register array (denoted by ▼) which stores $\mathbf{A}^{(j)}$ in the upper stage and passes their entries to a 3×3 register array (denoted as □) which computes the $\mathbf{A}^{(j+1)}$ matrix. What we don't show in the diagram is the transformation from $\mathbf{B}^{(j)}$ to $\mathbf{A}^{(j+1)}$, which involves (if the a_{11} in $\mathbf{B}^{(j)}$ is zero) the permutation of the first row with a non-zero leading element in $\mathbf{B}^{(j)}$ with the first row in $\mathbf{B}^{(j)}$, such that the element of the first row and the first column in $\mathbf{A}^{(j+1)}$ is nonzero. Notice that each stage in the matrix determinant calculation takes only one multiplication and one subtraction (omit the permutation time here). For the determinant calculation of a $t \times t$ matrix, it requires, at most, t stages.

It is obvious that Step 1 of the new algorithm is faster than Steps 1, 2 and 3 of the algorithm of Davida *et al.* in computing the number of errors in the received word. Now, we compare Steps 4 and 5 of the algorithm of Davida *et al.* to Step 2 of the new algorithm. One has to sum up $e!$ terms of product of t terms, if the Laplace expansion method [7] is used, to compute the determinant of the matrix $\mathbf{M}_{e+1}$ at Step 4 of their algorithm. And it takes n multiplications to finish the Chien search at Step 5. However, at Step 2 of the new algorithm, n determinants have to be evaluated in a pipelined way, in which each step requires one multiplication and one addition time. Thus, it requires $(n + e)$ multiplications and additions in total, in the worst case, at Step 2 of the new scheme.

To summarize this discussion, while Step 1 of the new decoding algorithm provides more efficient decoding speed than Steps 1, 2 and 3 of Davida *et al.*'s algorithm, Step 2 of the new

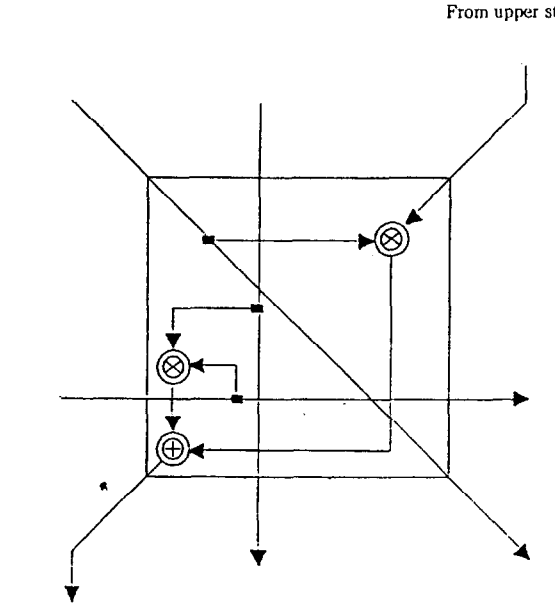

Figure 1. The cell of the pipeline determinant calculator.

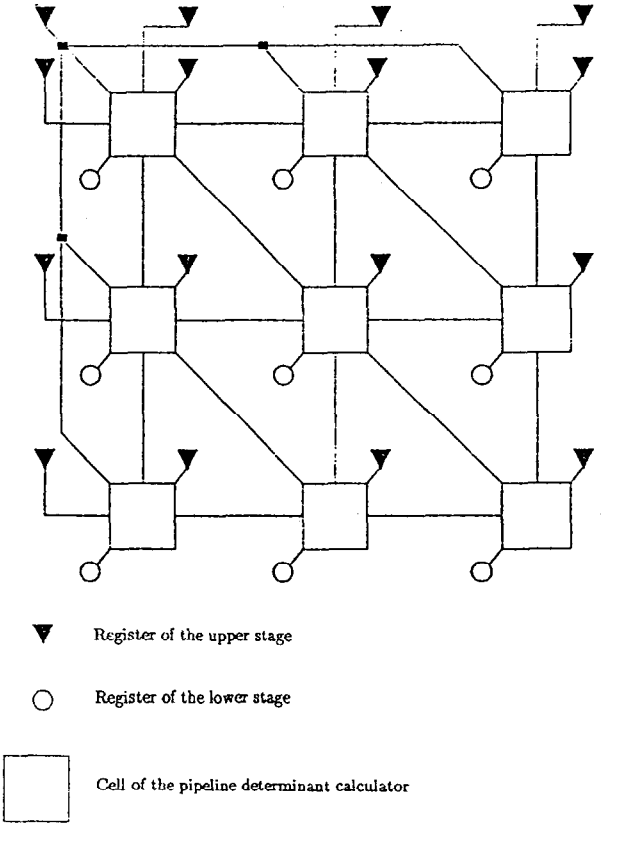

Figure 2. The interconnection diagram of two stages.

method allows us to decode in a pipelined way and, thus, can simplify the hardware design of the decoder.

REFERENCES

1. S. Lin and D.J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, (1983).
2. W.W. Peterson and E.J. Weldon, Jr., *Error-Correcting Codes*, The M.I.T. Press, Cambridge, MA, (1972).
3. E.R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, (1968).
4. G.I. Davida and J.W. Cowles, A new error-locating polynomial for decoding of BCH codes, *IEEE Trans. Inform. Theory* **IT-21** (2), 235–236 (March 1975).
5. J.L. Massey, Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes, *IEEE Trans. Inform. Theory* **IT-11** (4), 580–585 (October 1965).
6. H. Eves, *Elementary Matrix Theory*, Boston, (1966).
7. S.R. Searle, *Matrix Algebra Useful for Statics*, Wiley, New York, (1982).