Transactions Letters

New Serial Architecture for the Berlekamp–Massey Algorithm

Hsie-Chia Chang and C. Bernard Shung

Abstract— **We propose a new efficient serial architecture to implement the Berlekamp–Massey algorithm, which is frequently used in BCH and Reed–Solomon decoders. An inversionless Berlekamp–Massey algorithm is adopted which not only eliminates the finite-field inverter but also introduces additional parallelism. We discover a clever scheduling of *three* finite-field multipliers to implement the algorithm very efficiently. Compared to a previously proposed serial Berlekamp–Massey architecture, our technique significantly reduces the latency.**

***Index Terms*—Bose–Chaudhuri–Hacquenghem (BCH).**

I. Introduction

AMONG the most well-known error-correcting codes, the Bose–Chaudhuri–Hacquenghem (BCH) codes [1], [2], and the Reed–Solomon (RS) codes [3] are undoubtedly the most widely used block codes in communications and storage systems. For a comprehensive review of BCH and RS decoders, the texts by Berlekamp [4], Lin and Costello [5], or Blahut [6] are the best sources.

The most popular RS decoder architecture today can be summarized into four steps: 1) calculating the *syndromes* from the received codeword; 2) computing the *error locator polynomial* and the *error evaluator polynomial*; 3) finding the error locations; and 4) computing error values. The second step in the four-step procedure involves solving the *key equation* [4], which is[1]

$$S(x)\sigma(x) = \Omega(x) \bmod x^{2t}$$

where $S(x)$ is the syndrome polynomial, $\sigma(x)$ is the error locator polynomial and $\Omega(x)$ is the error evaluator polynomial.

The techniques frequently used to solve the key equation include the Berlekamp–Massey algorithm [4], [8], the Euclidean algorithm [9], and the continuous-fraction algorithm [10]. Compared to the other two algorithms, the Berlekamp-Massey algorithm is generally considered to be the one with the least hardware complexity [11]. Another advantage of the Berlekamp-Massey algorithm is that it can be formulated to compute $\sigma(x)$ only, thus saving a portion of the hardware used to compute $\Omega(x)$.

Existing architectures to implement the Berlekamp–Massey algorithm in hardware were proposed by Berlekamp [12], Liu [13], and Oh and Kim [14]. These proposals require $2t \sim 3t$ finite-field multipliers (FFM's) where t is the number of correctable errors. In addition, they all require a finite-field inverter (FFI) to implement the division operation, which imposes a significant hardware complexity. An inversionless Berlekamp–Massey algorithm was proposed by Burton [15] for BCH decoders, and was implemented by Reed, Shih, and Truong [11] for BCH and RS codes. However, more FFM's are required in the existing implementation of the inversionless Berlekamp–Massey algorithm [11].

In this letter we present a new architecture to implement the Berlekamp–Massey algorithm with drastically reduced hardware complexity while maintaining the overall decoding speed. Our work was motivated from the following observations. First, the existing architectures to implement the Berlekamp–Massey algorithm are *too fast*. Indeed, in the four-step decoding approach, the throughput is limited by syndrome calculation and Chien Search, each taking N cycles to finish, while existing architectures for the Berlekamp–Massey algorithm take $4t \sim 6t$ cycles to finish. Slowing down the Berlakamp–Massey algorithm till taking N cycles will not slow down the decoding. Therefore, we exploit *time sharing* a smaller number of FFM's to implement the Berlekamp–Massey algorithm. In this letter, such an approach will be termed a *serial* architecture, as in contrast to those *parallel* architectures [11]–[14].

To our knowledge, a serial architecture for the Berlekamp–Massey algorithm was firstly shown in a text by Blahut [6]. That architecture uses three FFM's and one FFI, and requires $2(t+1)$ clock cycles in each iteration. The clock cycle is determined by the the logic circuit delay of one FFM and one FFI. In this letter, we propose a new serial architecture which uses three FFM's and no FFI, and requires no more than $t+1$ clock cycles in each iteration. The clock cycle in our architecture is determined by the logic circuit delay of one FFM. Our architecture is therefore much faster than that in [6].

In Section II, we describe the time-sharing idea in details and present our efficient scheduling. In Section III, we show how to reconfigure the architecture to compute $\Omega(x)$. In Section IV we conclude the paper.

Paper approved by M. Fossorier, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received August 1, 1997; revised June 1, 1998 and October 7, 1998. This paper was presented at the 1998 International Conference on Communications, Atlanta, GA USA.

H.-C. Chang is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

C. B. Shung is with Allayer Technologies Corporation, San Jose, CA 95134 USA (e-mail: shung@allayer.com).

Publisher Item Identifier S 0090-6778(99)03316-4.

[1]In fact, the key equation defined in [4] was $(1 + S(x))\sigma(x) = \Omega(x) \bmod x^{2t}$, where the syndrome polynomial was defined to be $S(x) = \Sigma\, S_j x^j$. In our notation which follows [7] $S(x) = \Sigma\, S_j x^{j-1}$, and hence our key equation formulation is slightly different.

TABLE I
DATA DEPENDENCY OF THE INVERSIONLESS BERLEKAMP–MASSEY ALGORITHM AFTER DECOMPOSITION

cycle	$\Delta^{(i+1)}$	$\sigma^{(i)}(x)$
$j=0$	$\Delta^{(i)} = \Delta^{(i)}_{\nu_{i-1}} + S_{i-\nu_{i-1}+1}\sigma^{(i-1)}_{\nu_{i-1}}$	
	$\Delta^{(i+1)}_0 = 0$	$\sigma^{(i)}_0 = \delta\cdot\sigma^{(i-1)}_0$
$j=1$	$\Delta^{(i+1)}_1 = S_{i+2}\sigma^{(i)}_0$	$\sigma^{(i)}_1 = \delta\cdot\sigma^{(i-1)}_1 + \Delta^{(i)}\cdot\tau^{(i-1)}_0$
$j=2$	$\Delta^{(i+1)}_2 = \Delta^{(i+1)}_1 + S_{i+1}\sigma^{(i)}_1$	$\sigma^{(i)}_2 = \delta\cdot\sigma^{(i-1)}_2 + \Delta^{(i)}\cdot\tau^{(i-1)}_1$
$\vdots$	$\vdots$	$\vdots$
$j=\nu_i$	$\Delta^{(i+1)}_{\nu_i} = \Delta^{(i+1)}_{\nu_i-1} + S_{i-\nu_i+3}\sigma^{(i)}_{\nu_i-1}$	$\sigma^{(i)}_{\nu_i} = \delta\cdot\sigma^{(i-1)}_{\nu} + \Delta^{(i)}\cdot\tau^{(i-1)}_{\nu_i-1}$

II. EFFICIENT SCHEDULING OF SERIAL BERLEKAMP–MASSEY ARCHITECTURE

An inversionless Berlekamp–Massey algorithm is adopted in our architecture that is a $2t$-step iterative algorithm as shown in the following:

```
Initial condition :
    $D^{(-1)} = 0, \quad \delta = 1$
    $\sigma^{(-1)}(x) = \tau^{(-1)}(x) = 1$
    $\Delta^{(0)} = S_1$
for    $i = 0$  to  $2t-1$
    $\sigma^{(i)}(x) = \delta\cdot\sigma^{(i-1)}(x) + \Delta^{(i)}x\tau^{(i-1)}(x)$
    $\Delta^{(i+1)} = S_{i+2}\sigma^{(i)}_0 + S_{i+1}\sigma^{(i)}_1 + \cdots + S_{i-\nu_i+2}\sigma^{(i)}_{\nu_i}$
    If  $\Delta^{(i)} = 0$  or  $2D^{(i-1)} \geq i+1$
        $D^{(i)} = D^{(i-1)}, \quad \tau^{(i)}(x) = x\tau^{(i-1)}(x);$
    else
        $D^{(i)} = i+1-D^{(i-1)}, \quad \delta = \Delta^{(i)}$
        $\tau^{(i)}(x) = \sigma^{(i-1)}(x)$
```

where $\sigma^{(i)}(x)$ is the ith step error locator polynomial with degree ν_i, and $\sigma^{(i)}_j$'s are the coefficients of $\sigma^{(i)}(x)$; $\Delta^{(i)}$ is the ith step discrepancy and δ is a previous discrepancy; $\tau^{(i)}(x)$ is an auxiliary polynomial and $D^{(i)}$ is an auxiliary degree variable.

Define

$$\sigma^{(i)}_j = \begin{cases} \delta\cdot\sigma^{(i-1)}_0, & \text{for } j=0 \\ \delta\cdot\sigma^{(i-1)}_j + \Delta^{(i)}\,\tau^{(i-1)}_{j-1}, & \text{for } 1\leq j\leq\nu_i \end{cases}$$

$$\Delta^{(i+1)}_j = \begin{cases} 0, & \text{for } j=0 \\ \Delta^{(i+1)}_{j-1} + S_{i-j+3}\cdot\sigma^{(i)}_{j-1}, & \text{for } 1\leq j\leq\nu_i \end{cases}$$

where $\sigma^{(i)}(x) = \sigma^{(i)}_0 + \sigma^{(i)}_1 x + \cdots + \sigma^{(i)}_{\nu_i}x^{\nu_i}$, $\tau^{(i)}_j$'s are the coefficients of $\tau^{(i)}(x)$, and $\Delta^{(i+1)}_j$'s are the *partial results* in computing $\Delta^{(i+1)}$. At cycle 0 of $(i+1)$th step, we get

$$\begin{aligned}\Delta^{(i+1)} &= \Delta^{(i+1)}_{\nu_i} + S_{i-\nu_i+2}\sigma^{(i)}_{\nu_i} \\ &= \cdots \\ &= S_{i+2}\sigma^{(i)}_0 + S_{i+1}\sigma^{(i)}_1 + \cdots + S_{i-\nu_i+2}\sigma^{(i)}_{\nu_i}\end{aligned}$$

In other words, we can *decompose* the ith iteration into $\nu_i + 1$ *cycles* ($\nu_i \leq t$). In each cycle $\sigma^{(i)}_j$ requires at most two finite-field multiplications and $\Delta^{(i+1)}_j$ requires only one finite-field multiplication. The data dependency of the decomposed algorithm can be seen in Table I.

It is evident from Table I that, at cycle j, the computation of $\Delta^{(i+1)}_j$ requires $\sigma^{(i)}_{j-1}$ and $\Delta^{(i+1)}_{j-1}$, which have been computed at cycle $(j-1)$. Similarly, at cycle j, the computation of

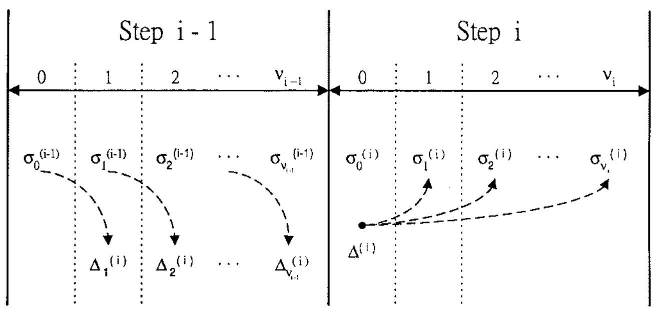

Fig. 1. The scheduling and data dependency of the decomposed inversionless Berlekamp–Massey algorithm. The dotted line represents the data dependency.

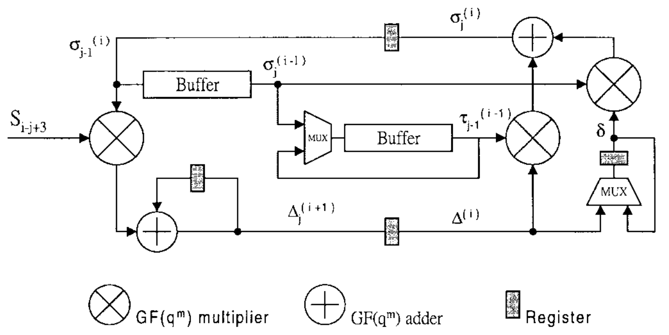

Fig. 2. The three-FFM architecture for implementing the inversionless Berlekamp–Massey algorithm.

$\sigma^{(i)}_j$ requires $\Delta^{(i)}$ and $\sigma^{(i-1)}_j$, which have been computed at cycle 0 and the $(i-1)$th step, respectively. Note that the original Berlekamp–Massey algorithm cannot be scheduled as efficiently because the computation of $\sigma^{(i)}_j$ requires two *sequential* multiplications and one inversion. The inversionless Berlekamp–Massey algorithm provides the necessary parallelism to allow our efficient scheduling. The scheduling and data dependency of the decomposed algorithm are further illustrated in Fig. 1.

The decomposed algorithm shown above suggested a three-FFM implementation of the inversionless Berlekamp–Massey algorithm, which is shown in Fig. 2. Though not shown in this letter, our architecture can also be used in the correction of both errors and erasures. Compared to the previously proposed parallel architectures [11]–[14], our architecture reduces the hardware complexity significantly. Compared to a previously proposed serial architecture [6], our architecture reduces the time complexity significantly because of (1) the reduction of the number of clock cycles, and (2) the reduction of cycle time. Therefore, the proposed architecture achieves an optimization in the area-delay product.

III. EFFICIENT COMPUTATION OF $\Omega(x)$

The conventional way to compute the error evaluator polynomial, $\Omega(x)$, is to do it in parallel with the computation of $\sigma(x)$. Using the Berlekamp–Massey algorithm, this involves an iterative algorithm to compute $\Omega^{(i)}(x), i = 0, \cdots, 2t-1$. However, if $\sigma(x)$ (with degree ν) is first obtained, we have

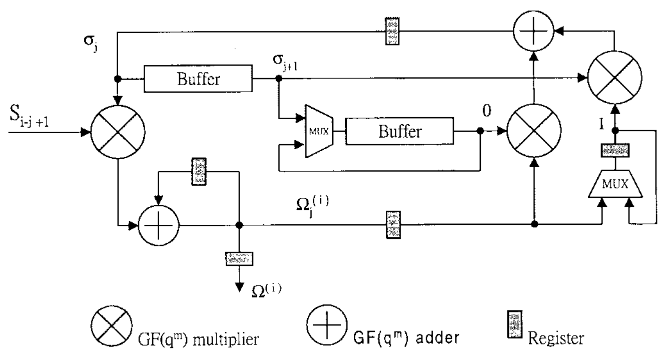

Fig. 3. The three-FFM architecture is reconfigured to compute $\Omega(x)$. Note the labels in this figure are different from those in Fig. 2.

from the key equation and the Newton's identity

$$\begin{aligned}\Omega(x) &= S(x)\sigma(x) \bmod x^{2t}\\ &= (S_1 + S_2x + \cdots + S_{2t}x^{2t-1})\\ &\quad \cdot (\sigma_0 + \sigma_1 x + \cdots + \sigma_\nu x^\nu) \bmod x^{2t}\\ &\triangleq \Omega_0 + \Omega_1 x + \cdots + \Omega_{t-1}x^{t-1}\\ \Omega_i &= S_{i+1}\sigma_0 + \cdots + S_1\sigma_i, \qquad i = 0, 1, \cdots, t-1.\end{aligned}$$

That is, the computation of $\Omega(x)$ can be performed *directly* after $\sigma(x)$ is computed. Note that the direct computation requires fewer multiplications than the iterative algorithm which computes many *unnecessary* intermediate results. The penalty of this efficient computation is the additional latency because $\sigma(x)$ and $\Omega(x)$ are computed in sequence.

Furthermore, it can be seen that the computation of Ω_i is very similar to that of $\Delta^{(i)}$ except some minor differences. Therefore, the same hardware used to compute $\sigma(x)$ can be *reconfigured* to compute $\Omega(x)$ after $\sigma(x)$ is computed. Like $\Delta^{(i)}$, we compute Ω_i as follows:

$$\Omega_i^{(j)} = \begin{cases} S_{i+1}\sigma_0, & \text{for } j = 0\\ \Omega_i^{j-1} + S_{i-j+1}\sigma_j, & \text{for } 1 \le j \le i\end{cases}$$

In Fig. 3, we show how the same three-FFM architecture can be reconfigured to compute $\Omega(x)$.

IV. Conclusion

In this letter we propose a new efficient serial architecture to implement the Berlekamp–Massey algorithm, which is frequently used in BCH and RS decoders. An inversionless Berlekamp–Massey algorithm is adopted which not only eliminates the FFI, but also introduces additional parallelism to the computation. We discover a clever scheduling of three FFM's to implement the algorithm very efficiently. To efficiently compute $\Omega(x)$, the computation is performed after $\sigma(x)$ is obtained. Moreover, in our architecture the computation of $\Omega(x)$ and $\sigma(x)$ shares the same hardware. Our technique can also be applied to the correction of both errors and erasures. Compared to the previously proposed parallel Berlekamp-Massey architectures, our architecture significantly reduces the hardware complexity. Compared to a previously proposed serial Berlekamp–Massey architecture, our architecture significantly reduces the timing complexity. Therefore, our architecture achieves an optimization in the area-delay product.

References

[1] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inform. Contr.*, vol. 3, pp. 68–69, 1960.
[2] A. Hocquenghem, "Codes corecteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, 1959.
[3] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, pp. 300–304, 1960.
[4] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[5] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
[6] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.
[7] T. K. Truong, W. L. Eastman, I. S. Reed, and I. S. Hsu, "Simplified procedure for correcting both errors and erasures of Reed-Solomon code using Euclidean algorithm," *Proc. Inst. Elect. Eng. pt. E*, vol. 135, no. 6, pp. 318–324, 1988.
[8] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, 1969.
[9] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inform. Contr.*, vol. 27, pp. 87–99, 1975.
[10] L. R. Welch and R. A. Scholtz, "Continued fractions and Berlekamp's algorithm," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 19–27, 1979.
[11] I. S. Reed, M. T. Shih, and T. K. Truong, "VLSI design of inverse-free Berlekamp–Massey algorithm," *Proc. Inst. Elect. Eng. pt. E*, vol. 138, pp. 295–298, Sept. 1991.
[12] E. R. Berlekamp, *Galois Field Computer*, U.S. Patent 4162480, 1979.
[13] K. Y. Liu, "Architecture for VLSI design of Reed-Solomon decoders," *IEEE Trans. Computers*, vol. C-33, pp. 178–189, Feb. 1984.
[14] Y. U. Oh and D. Y. Kim, "Method and apparatus for computing error locator polynomial for use in a Reed–Solomon decoder," U.S. Patent 4 663 470, 1996.
[15] H. O. Burton, "Inversionless decoding of binary BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 464–466, 1971.