Binary GCD algorithm for computing error locator polynomials in Reed-Solomon decoding

F. Argüello

The binary GCD algorithm, discovered by Stein, is an alternative to the Euclidean algorithm for computing the greatest common divisor of two integers. In this work, the binary GCD algorithm is applied to Reed-Solomon decoding and a novel iterative algorithm for computing error locator polynomials is proposed. Compared to Euclidean-based algorithms, this algorithm exhibits some speed and area advantages.

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

$$S(x)\Lambda(x) = \Omega(x) \bmod x^{2t} \qquad (1)$$

where $S(x)$ is the syndrome polynomial, $\Lambda(x)$ is the error locator polynomial and $\Omega(x)$ is the error evaluator polynomial. The techniques more frequently used to solve the key equation are the Berlekamp-Massey algorithm and the Euclidean algorithm [1, 2].

Most of the iterative greatest common divisor (GCD) algorithms can be categorised as the derivatives of two basic algorithms proposed by Euclid and Stein, respectively, around 250 BC and in the 1960s [3, 4]. The binary GCD algorithm (Stein's algorithm) uses the following observations:

– If a and b are both even, $\gcd(a, b) = 2\gcd(a/2, b/2)$.
– If a is even and b is odd, $\gcd(a, b) = \gcd(a/2, b)$.
– Otherwise both are odd, and $\gcd(a, b) = \gcd(|a - b|/2, b)$.

The binary GCD algorithm has been used as an alternative to the Euclidean algorithm in several applications. In [5], Takagi successfully extended Stein's algorithm and developed a set of modular division algorithms over Z_p for odd prime p. Recently, Watanabe, Takagi and Takagi [6] successively applied Stein's algorithm to GF(2^m) and Wu *et al.* [7] developed a set of modular division algorithms with guaranteed convergence in $2m - 1$ iterations. In this Letter we propose a binary GCD-based algorithm for obtaining the error location polynomial in RS decoding with convergence in $2t$ iterations.

Derivation of algorithm: A minimum-degree error locator polynomial $\Lambda(x)$ is defined for the second step in decoding RS codes. The binary GCD algorithm in its original form is especially efficient for operations on binary representations. Moreover, it can be extended to apply to polynomial representations.

Let $a(x)$ be a polynomial over the finite field GF($2m$):

$$a(x) = \sum_{i=0}^{n-1} a_i x^i, \quad a_i \in GF(2^m) \qquad (2)$$

and its associated vector representation is denoted by $a = [a_{n-1}; \ldots\ a_1, a_0]$. We can use the following observations:

– If $a(x)$ and $b(x)$ are both 'even', $\gcd(a, b) = x\gcd(a/x, b/x)$.
– If $a(x)$ is 'even' and $b(x)$ is 'odd', $\gcd(a, b) = \gcd(a/x, b)$.
– Otherwise both are 'odd', and $\gcd(a, b)$–$\gcd([a + (a_0/b_0)b] = x, b)$.

By abuse of terminology, we say that a polynomial $a(x)$ is 'even' when $a_0 = 0$ and 'odd' otherwise.

Using the above observations we can apply the binary GCD method developed in [6, 7] to the RS decoding. The method is based on the application of successive linear transformations in two pairs of polynomials: $[r(x), s(x)]$ and $[u(x), v(x)]$, starting with $[r(x), s(x)] = [S(x), x^{2t}]$ ($S(x)$ is the syndrome polynomial) and $[u(x), v(x)] = [1, 0]$. As in the binary algorithm for modular inversion in GF(2^m) [7], two iterative steps are computed in each linear transformation. The first iterative step is a bit-wise XOR operation of pairs and a reassignment (an Euclidean-based algorithm would have an additional step of alignment of the polynomials $r(x)$ and $s(x)$). In the second iterative step, we remove powers of x from $r(x)$ and $u(x)$. These two iterative steps can be easily obtained from the above observations. This recursive computation is performed through exactly $2t$ iterations (an Euclidean-based algorithm would stop when degree$(r) < t$). It is similar to that of the binary GCD algorithm for computing multiplicative inverses in GF(2^m), which requires exactly $2m - 1$ iterations. When the algorithm terminates, we obtain the error locator polynomial from the variable $u(x)$.

Algorithm: Binary GCD-based Reed-Solomon decoder
Input: $S(x)$ (Syndrome polynomial)
Output: $\Lambda(x) = u(x)$ (Error locator polynomial)

```
 1.  Initialise $[r(x), s(x)] = [S(x), x^{2t}]$; $[u(x), v(x)] = [1, 0]$; $\delta = -1$;
 2.  For ($i$–1; $i \le 2t$; i++)
 3.  {   If ($r(x)$, is 'odd')
 4.      {   If($s(x)$ is 'odd') $\rho = r_0/s_0$;   Else $\rho = 1$;
 5.          If ($\delta \ge 0$) { $r(x) = r(x) + \rho s(x)$; $u(x) = u(x) + \rho v(x)$, }
 6.          Else {$[r(x), s(x)] = [r(x) + \rho s(x), r(x)]$;
                 $[u(x), v(x)] = [u(x) + \rho v(x), u(x)]$;
 7.              $\delta = -\delta$;}
         }
 8.      $u_{2t} = u_0$;   /* $u(x) = u(x) + u_0 x^{2t}$ */
 9.      $r(x) = r(x) \gg 1$; $u(x) = u(x) \gg 1$;
10.      $\delta = \delta - 1$;}
```

Example: The algorithm will be illustrated for the simple example of a (7, 3) RS code over GF(2^3) using the primitive polynomial $\alpha^3 + \alpha + 1$. In this case, $n = 7$, $k = 3$, $t = 2$ and the generator polynomial is $G(x) = [1, 3, 1, 2, 3]$ (the first one is the high order coefficient). Assume $T(x) + E(x) = [6, 7, 4, 7, 7, 0, 4]$ be the received sequence (with errors) which generates the syndrome polynomial $S(x) = [3, 5, 3, 2]$. Next, this syndrome is used as input to the algorithm.

Table 1 shows the values of the variables in the successive iterations during the execution of the algorithm. When the algorithm terminates we obtain the error locator polynomial, $\Lambda(x) = u(x) = [5, 5, 1]$. Additionally, we can compute the error evaluator polynomial $\Lambda(x) = [2, 2]$, the error polynomial $E(x) = [0, 2, 0, 0, 0, 6, 0]$, and the transmitted sequence $T(x) = [6, 5, 4, 7, 7, 6, 4]$ (the data sequence is [6, 5, 4]).

Table 1: Example of execution of algorithm

i	$r(x)$	$s(x)$	$u(x)$	$v(x)$	δ
	[0, 3, 5, 3, 2]	[1, 0, 0, 0, 0]	[0, 0, 0, 0, 1]	[0, 0, 0, 0, 0]	–1
1	[0, 1, 3, 5, 3]	[0, 3, 5, 3, 2]	[0, 1, 0, 0, 0]	[0, 0, 0, 0, 1]	0
2	[0, 0, 6, 1, 2]	[0, 3, 5, 3, 2]	[0, 4, 1, 0, 0]	[0, 0, 0, 0, 1]	–1
3	[0, 0, 3, 3, 2]	[0, 0, 6, 1, 2]	[0, 1, 4, 1, 0]	[0, 4, 1, 0, 0]	0
4	[0, 0, 0, 5, 2]	[0, 0, 6, 1, 2]	[0, 0, 5, 5, 1]	[0, 4, 1, 0, 0]	–1

Evaluation: Compared to the computation of error locator polynomials based on Euclid's algorithm [1, 2], the binary GCD-based one requires a similar number of AND and XOR operations for updating $r(x)$, $s(x)$, $u(x)$ and $v(x)$, but exhibits the following advantages. Computation of degrees of polynomials is not necessary because all multiplicative factors are polynomial coefficients of degree zero (in Euclidean-based algorithms they are leading polynomial coefficients) and there are not comparisons of polynomials (in Euclidean-based algorithms there are polynomial degree comparisons between $r(x)$ and $s(x)$). The binary GCD-based algorithm stops in a fixed number of stages (Euclidean-based algorithms stop when degree$(r) < t$). And it is not necessary to perform alignments of polynomials.

By eliminating these complicated operations, important area and time savings can be achieved. Moreover, the algorithm can easily be modified to be inversion-free (defining $[\rho, \eta] = [r_0, s_0]$ if $s(x)$ is 'odd' or $[\rho, \eta] = [1, 1]$ if $s(x)$ is 'even', and writing $r(x) = \eta r(x) + \rho s(x)$; $u(x) = \eta u(x) + \rho v(x)$).

Conclusions: We have developed a binary GCD-based algorithm for computing the error locator polynomials in Reed-Solomon decoding with convergence in $2t$ iterations. The presented algorithm exhibits the following advantages. Computation of degrees of polynomials is not necessary (multiplicative factors are polynomial coefficients of degree zero), the algorithm stops in a fixed number of stages (not based on polynomials' degrees), and it is not necessary to perform alignments of polynomials. Also, it can be formulated to be inversion-free.

Acknowledgments: This work was supported in part by the Xunta de Galicia under contract PGIDIT03TIC10502PR and by MCYT under contract TIN2004-07797-C02.

© IEE 2005 *2 March 2005*
Electronics Letters online no: 20050769
doi: 10.1049/el:20050769

F. Argüello (*Department of Electronics and Computer Science, University of Santiago, 15782 Santiago de Compostela, Spain*)

E-mail: arguello@dec.usc.es

References

1 Lee, H.: 'High-speed VLSI architecture for parallel Reed-Solomon decoder', *IEEE Trans. VLSI Syst.*, 2003, **11**, (2), pp. 288–295

2 Lee, S.S., and Song, M.K.: 'An efficient recursive cell architecture of modified Euclid's algorithm for decoding Reed-Solomon codes', *IEEE Trans. Consum. Electron.*, 2002, **48**, (4), pp. 845–849

3 Stein, J.: 'Computational problems associated with Racah algebra', *J. Comput. Phys.*, 1967, **1**, pp. 397–405

4 Knuth, D.E.: 'The art of the computer programming: seminumerical algorithms' (Addison-Wesley, Reading, MA, 1981)

5 Takagi, N.: 'A VLSI algorithm for modular division based on the binary GCD algorithm', *IEICE Trans. Fundam.*, 1998, **E81-A**, (5), pp. 724–728

6 Watanabe, Y., Takagi, N., and Takagi, K.: 'A VLSI algorithm for division in GF(2^m) based on extended binary GCD algorithm', *IEICE Trans. Fundam.*, 2002, **E85-A**, (5), pp. 994–999

7 Wu, C.-H., Wu, C.-M., Shieh, M.-D., and Hwang, Y.-T.: 'High-speed, low-complexity systolic designs of novel iterative division algorithms in GF(2^m)', *IEEE Trans. Comput.*, 2004, **53**, (3), pp. 375–380