US 20030126543A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0126543 A1

Lee et al. (43) Pub. Date: Jul. 3, 2003

(54) METHOD AND APPARATUS FOR SOLVING KEY EQUATION POLYNOMIALS IN DECODING ERROR CORRECTION CODES

(76) Inventors: **Chen-Yi Lee**, Hsinchu (TW); **Hsie-Chia Chang**, Hsinchu (TW)

Correspondence Address:
PERKINS COIE LLP
P.O. BOX 2168
MENLO PARK, CA 94026 (US)

(21) Appl. No.: **10/155,488**

(22) Filed: **May 22, 2002**

(30) **Foreign Application Priority Data**

Nov. 28, 2001 (TW)..... 090129778

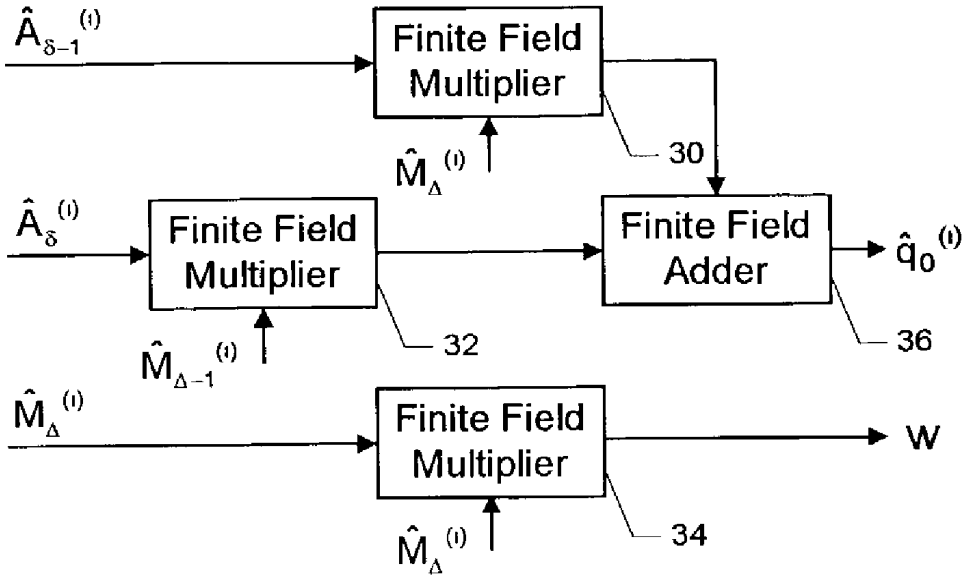
Publication Classification

(51) **Int. Cl.**[7] **H03M 13/00**

(52) **U.S. Cl.** **714/784**

(57) **ABSTRACT**

The presently invention discloses a method for computing error locator polynomial and error evaluator polynomial in the key equation solving step of the error correction code decoding process whereby the polynomials are generated through at most t intermediate iterations that can be implemented with minimal amount of hardware circuitry. However, depending on the selected (N,K) code, the number of cycles required for the calculation of the polynomials would be within the time required for the calculation of upstream data. Additionally, the present invention for computing the error locator polynomial and the error value polynomial employs an efficient scheduling of a small number of registers and finite-field multipliers (FFMs) without the need of finite-field inverters (FFIs) is illustrated. Using these new methods, a new area-efficient architecture that uses only $4t+2\rho+4$ registers and three FFMs and no FFIs is presented to implement the inversionless Euclidean algorithm.



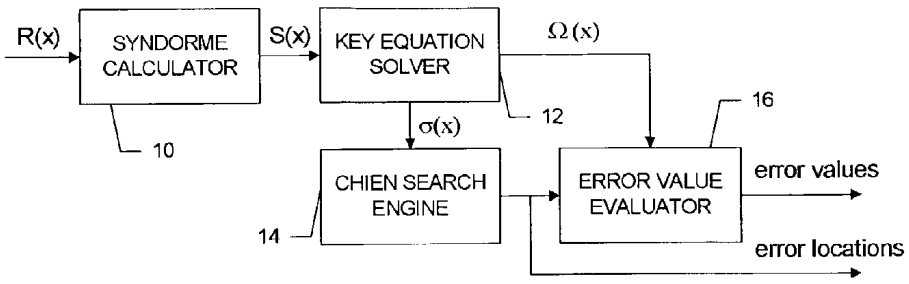

R(x)
SYNDORME CALCULATOR
S(x)
KEY EQUATION SOLVER
Ω(x)
16
σ(x)
12
10
CHIEN SEARCH ENGINE
ERROR VALUE EVALUATOR
error values
14
error locations

Fig.1a

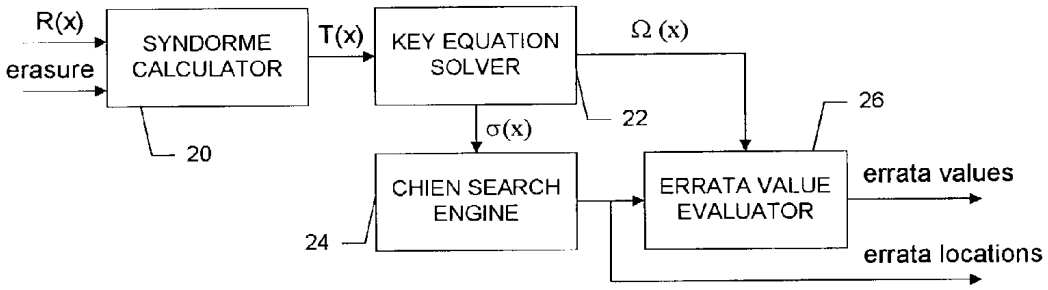

R(x)
erasure
SYNDORME CALCULATOR
T(x)
KEY EQUATION SOLVER
Ω(x)
22
26
σ(x)
20
CHIEN SEARCH ENGINE
ERRATA VALUE EVALUATOR
errata values
24
errata locations

Fig.1b

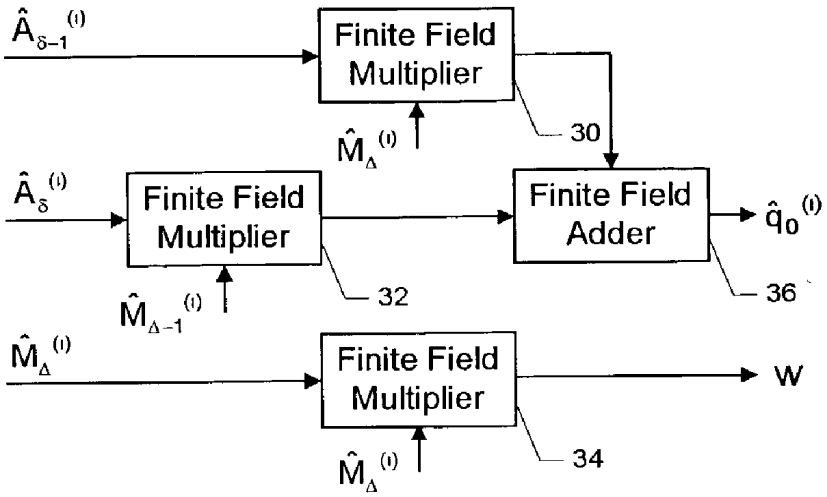

$\hat{A}_{\delta-1}^{(i)}$
Finite Field Multiplier
30
$\hat{M}_{\Delta}^{(i)}$
$\hat{A}_{\delta}^{(i)}$
Finite Field Multiplier
Finite Field Adder
$\hat{q}_0^{(i)}$
32
36
$\hat{M}_{\Delta-1}^{(i)}$
$\hat{M}_{\Delta}^{(i)}$
Finite Field Multiplier
W
34
$\hat{M}_{\Delta}^{(i)}$

Fig.2a

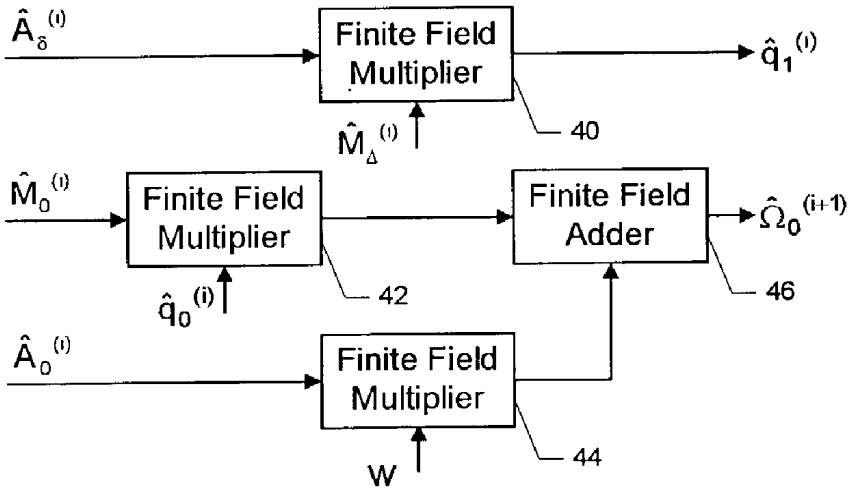

$\hat{A}_{\delta}^{(i)}$
Finite Field Multiplier
$\hat{q}_1^{(i)}$
40
$\hat{M}_{\Delta}^{(i)}$
$\hat{M}_0^{(i)}$
Finite Field Multiplier
Finite Field Adder
$\hat{\Omega}_0^{(i+1)}$
42
46
$\hat{q}_0^{(i)}$
$\hat{A}_0^{(i)}$
Finite Field Multiplier
44
W

Fig.2b

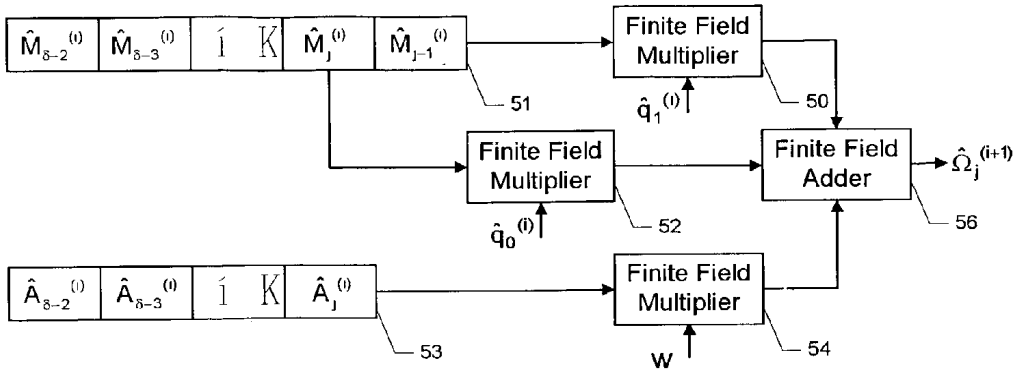


Fig.2c

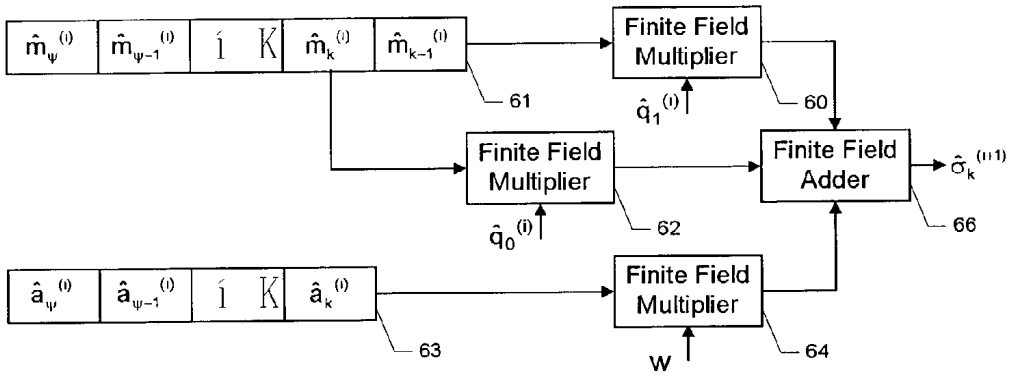


Fig.2d

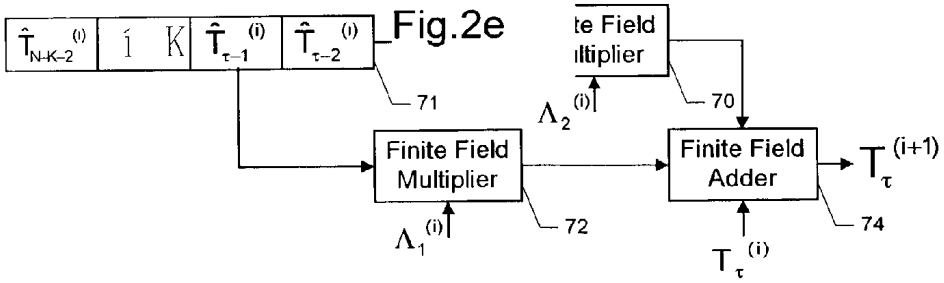


Fig.2e

METHOD AND APPARATUS FOR SOLVING KEY EQUATION POLYNOMIALS IN DECODING ERROR CORRECTION CODES

BACKGROUND OF THE INVENTION

[0001] In the transmission of data from a source location to a destination location through a variety of media, noise caused by the transmission path and/or the media itself causes errors in the transmitted data. Thus, the data transmitted is not the same as the data received. In order to determine the errors in the received data, various methods and techniques have been developed to detect and correct the errors in the received data. One of the methods is to generate a codeword which includes a message part (data to be transmitted) and a parity part (information for performing error correction).

[0002] Among the most well-known error-correcting codes, the BCH (Bose-Chaudhuri-Hocquenghen) codes and the RS (Reed-Solomon) codes are the most widely used block codes in the communication field and storage systems. The mathematical basis of BCH and RS codes is explained by: E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, New York, 1968; and Richard E. Blahut, *Theory and Practice of Error Control Codes,* Addison-Wesley, 1983.

[0003] An (N, K) BCH or RS code has K message symbols and N coded symbols, where each symbol belongs to GF(q) for a BCH code or GF(q^m) for a RS code. A binary (N, K) BCH code can correct up to t errors with N=2m−1, N-K<=mt. An (N, K) RS code can correct up to t errors and ρ erasures with

$$t = \left\lfloor \frac{N-K-\rho}{2} \right\rfloor .$$

[0004] For binary BCH codes, an error can be corrected simply by finding out the error location. For RS codes, an error can be corrected by finding out the error location and the error value. In RS codes, an erasure is defined to be an error with a known error location, and hence its correction reduces to finding the error value.

[0005] The method steps for common popular RS decoder architectures for the correction of errors can be summarized into four steps: (1) calculating the syndromes from the received codewords, (2) computing the error locator polynomial and the error evaluator polynomial, (3) finding the error locations, and (4) computing error values. If both errors and erasures and corrected, the four steps are modified to: (1) calculating the Forney syndromes from the received codewords and the erasure locations, (2) computing the errata locator polynomial and the errata evaluator polynomial, (3) finding the errata locations, and (4) computing the errata values.

[0006] Referring to **FIG. 1**, the general decoding steps are illustrated. Note that for simplification, the error-only RS decoder is introduced. The received data, R(x), is provided to a syndrome generator **10,20** to generate a syndrome polynomial, S(x), representing the error pattern of the codeword from which the errors can be corrected. The syndrome is then provided to a key equation solver **12,22** to generate an error locator polynomial, σ(x), and an error evaluator polynomial, Ω(x). The error locator polynomial indicates the location(s) of the error and the error evaluator polynomial indicates the amount of the error. In the next step, the error locator polynomial is passed to a Chien search engine **14,24** to generate the root(s), β_1, representing the location(s) of the errors. Then the error evaluator **16,26** receiving the root(s) and the error evaluator polynomial, Ω(x), calculates the error value(s) of the root(s)

[0007] The second step in the above-mentioned four-step procedure involves solving the key equation, which is

$$S(x)\sigma(x)=\Omega(x) \bmod x^{N-K} \quad (1)$$

[0008] where S(x) is the syndrome polynomial, σ(x) is the error locator polynomial and Ω(x) is the error evaluator polynomial. When both errors and erasures are corrected, σ(x) and Ω(x) are the errata locator polynomial and the errata evaluator polynomial, respectively. In addition, the errata locator polynomial σ(x) becomes the product of λ(x) and Λ(x) corresponding to the error locator polynomial and the erasure locator polynomial, respectively.

[0009] The techniques frequently used to solve the key equation (1) include the Berlekamp-Massey algorithm and the Euclidean algorithm. The extension of these algorithms to correct both errors and erasures can be found in the Blahut article cited above. Here a novel invertionless decomposed Euclidean architecture is invented to reduce the hardware complexity drastically while maintaining the over all decoding speed.

[0010] Prior art technologies applied the traditional Euclidean algorithm (or variation thereof) for the calculation of the error locator polynomial and the error evaluator polynomial, and designed circuits based upon these algorithms. However, each of these algorithms require a large number of registers, finite-field multipliers (FFM) and perhaps a finite-field inverters (FFI). Each of the FFMs and FFI translates into a hardware circuitry and real estate on an integrated circuit chip. Therefore, the goal here is to derive a method for solving of the polynomials in an efficient manner and to minimize the amount of circuitry required in the implementation of the algorithm. The number of registers and FFMs is typically a function of the variable t. Table 1 illustrates the authors of the architectures for correcting error-only codewords and the corresponding number of registers, FFMs and FFI:

TABLE 1

Reference	Registers as a function of t	FFMs as a function of t	FFI
Reed	8t + 2	8t	0
Song	6t + 4	6t + 2	0
Wu	7t + 5	$t+\left\lceil \frac{t+1}{2} \right\rceil$	1

[0011] From Table 1, Reed proposed the implementation of the inversionless Euclidean algorithm requires 8t+2 registers, 8t FFMs and no FFI in *VISI Implementation of A Pipeline Reed-Solomon Decoder,* IEEE Transaction on Computers, vol. C-34, pp. 393-403, May 1985. In addition, the article *An Efficient Architecture for Implementing the Modified Euclidean Algorithm,* the 9th NASA Symposium

on VLSI Design, November 2000, Song demonstrated an architecture requiring 6t+4 registers and 6t+2 FFM's and no FFI.

[0012] On the other hand, the article *An Area-efficient Versatile Reed-Solomon Decoder for ADSL,* IEEE International Symposium on Circuits and Systems, May 1999, Wu et al. presented the architecture reducing the number of FFMs but requiring the relatively complex FFI, which will limit speed and impose a significant hardware complexity.

[0013] Therefore, it would be desirable to have an inversionless method and apparatus that requires no FFIs and minimizes the number of registers and FFMs in the implementation thereof.

SUMMARY OF THE INVENTION

[0014] Accordingly, it is an object of the present invention to provide a method and apparatus for solving key equation polynomials in the decoding of codewords. Based upon the Euclidean algorithm, it can be implemented with minimal hardware circuitry.

[0015] It is another object of the present invention to provide a method and apparatus for solving key equation within a t-step iterative decoding procedure while the prior art architectures require at most 2t iterations.

[0016] It is yet another object of the present invention to provide a method and apparatus for solving key equation polynomials without decreasing the overall decoding speed of the decoder.

[0017] Briefly, in a presently preferred embodiment, a method for computing error locator polynomial and error evaluator polynomial in the key equation solving step of the error correction code decoding process is presented whereby the polynomials are generated through at most t intermediate iterations that can be implemented with minimal amount of hardware circuitry. However, depending on the selected (N,K) code, the number of cycles required for the calculation of the polynomials would be within the time required for the calculation of upstream data.

[0018] Additionally, a presently preferred embodiment for computing the error locator polynomial and the error value polynomial employs an efficient scheduling of a small number of registers and finite-field multipliers (FFMs) without the need of finite-field inverters (FFIs) is illustrated. Using these new methods, a new area-efficient architecture that uses only $4t+2\rho+2$ registers and three FFMs and no FFIs is presented to implement the inversionless Euclidean algorithm. This method and architecture can be applied to a wide variety of RS and BCH codes with suitable code sizes.

[0019] More specifically, the 3-FFM architecture of the presently preferred embodiment for solving key equation polynomials can also be utilized to calculate the Forney syndrome polynomial T(x) described above. This method and architecture can be applied to correct the error-only as well as the error-and-erasure codewords.

[0020] An advantage of the present invention is that it provides a method and apparatus for solving key equation polynomials in the decoding of codewords. Based upon the Euclidean algorithm, it can be implemented with minimal hardware circuitry.

[0021] Another advantage of the present invention is that it provides a method and apparatus for solving key equation polynomials within a t-iteration decoding procedure while other architectures require at most 2t iterations. It will maintain the overall decoding speed of the decoder.

[0022] Yet another advantage of the present invention is that it provides an identical method and apparatus for not only solving key equation polynomials but also calculating the Forney syndrome polynomial T(x). It can be applied to the correction of errors as well as erasures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] **FIGS. 1***a* and **1***b* illustrates the processing blocks in the decoding or codewords;

[0024] **FIG. 2***a*~**FIG. 2***c* shows a three-FFM architecture of the preferred embodiment for calculating the errata evaluator polynomial, $\Omega(x)$, in the key equation solver.

[0025] **FIG. 2***d* shows a three-FFM architecture of the preferred embodiment for calculating the errata location polynomial, $\sigma(x)$, in the key equation solver.

[0026] **FIG. 2***e* shows a three-FFM architecture of the preferred embodiment for calculating the Forney syndrome, T(x).

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0027] Firstly, we will show our modified decoding procedure requiring at most t iterations while the previous decoding procedure requires at most 2t iteration. Following the inversionless Euclidean algorithm is illustrated and the errata value(s) and errata location(s) produced by $\hat{\Omega}(x)$ and $\hat{\sigma}(x)$ in our inversionless decoding procedure are identical to the errata value(s) and errata location(s) founded by $\Omega(x)$ and $\sigma(x)$ in the original algorithm. Secondly, we decompose the inversionless Euclidean algorithm for reducing the number of registers to $4t+2\rho+2$ and the number of FFMs to 3. Finally, we show the condition on N, K such that our architecture can be applied.

[0028] The Euclidean Decoding Procedure

[0029] For illustrating the Euclidean algorithm, we rewrite (1) as:

$$\Omega(x)=x^{N-K}Q(x)+T(x)\lambda(x) \tag{2}$$

[0030] where Q(x) is the quotient polynomial of x^{N-K} and $T(x)\lambda(x)$, $T(x)=S(x)\Lambda(x)$ is the Forney syndrome polynomial, and $\sigma(x)=\lambda(x)\Lambda(x)$ is the errata locator polynomial, which is the product of the error locator polynomial, $\lambda(x)$, and the erasure locator polynomial, $\Lambda(x)$. Therefore, the errata evaluator polynomial, $\Omega(x)$, can be calculated by the similar process of computing the GCD polynomial of x^{N-K} and T(x) through the Euclidean algorithm, whose decoding process can be shown as follows:

$$\begin{aligned}
&R^{(-1)}(x)=x^{N-K}\\
&R^{(0)}(x)=T(x)\\
&R^{(1)}(x)=R^{(-1)}(x)-R^{(0)}(x)\cdot Q^{(1)}(x)\\
&R^{(i)}(x)=R^{(i-2)}(x)-R^{(i-1)}(x)\cdot Q^{(i)}(x)
\end{aligned} \tag{3}$$

[0031] where $Q^{(i)}(x)$ is the i-th quotient polynomial and $R^{(i)}(x)$ is the i-th remainder polynomial. Each iterative step in (4) performs a polynomial division operation. Note that

the i-th dividend polynomial $R^{(i-2)}(x)$ and the i-th divisor polynomial $R^{(i-1)}(x)$ are equivalent to the (i−2)-th and the (i−1)-th remainder polynomials respectively. After n division operations, the n-th remainder polynomial, $R^{(n)}(x)$, is assumed to be the errata evaluator polynomial $\Omega(x)$. From the extended form of Euclidean algorithm introduced by *Error-Control Coding for Data Networks,* Kluwer Academic, 1999, the similar decoding process except the minor difference in the initial condition can be used to determine the errata locator polynomial $\sigma(x)$, which is also described as follows:

$$\mu^{(-1)}(x)=0$$
$$\mu^{(0)}(x)=\Lambda(x)$$
$$\mu^{(1)}(x)=\mu^{(-1)}(x)+\mu^{(0)}(x)\cdot Q^{(2)}(x)$$
$$\mu^{(i)}(x)=\mu^{(i-2)}(x)+\mu^{(i-1)}(x)\cdot Q^{(i)}(x) \quad (4)$$

[0032] where

$$\Lambda(x)=\prod_{\alpha^i\in\Lambda}(1+\alpha^i x)$$

[0033] represents the erasure locator polynomial and Λ is the erasure set. Note that all $Q^{(i)}(x)$ here are equivalent to the i-th quotient polynomial $Q^{(i)}(x)$ in (3). Similarly, after n iterations, $\mu^{(n)}(x)$ is assumed to be the errata locator polynomial, $\sigma(x)$. From (3) and (4), it can be shown that the sum of $\deg(R^{(i-1)}(x))$ and $\deg(\mu^{(i)}(x))$ equals to a constant number, N-K+s, where s is the number of actual erasures and hence, equals the degree of $\Lambda(x)$.

[0034] Our Modified Decoding Procedure

[0035] The proposed modified decoding procedure calculating the quotient polynomial with degree one in advance is shown as follows:

[0036] Initial Condition

$$A^{(0)}(x)=x^{N-K},\ M^{(0)}(x)=\Omega^{(0)}(x)=T(x)$$
$$a^{(0)}(x)=0,\ m^{(0)}(x)=\sigma^{(0)}(x)=\Lambda(x)$$

[0037] For(i=0 to t)

$$\delta=\deg(A^{(i)}(x)),\ \Delta=\deg(M^{(i)}(x))$$

[0038] if($\deg(\sigma^{(i)}(x))\leqq\deg(\Omega^{(i)}(x))$

$$q_1^{(i)}(x)=\frac{A_\delta^{(i)}}{M_\Delta^{(i)}}$$
$$q_0^{(i)}(x)=0 \quad \text{for } \delta=\Delta$$
$$q_0^{(i)}(x)=\frac{M_\Delta^{(i)}A_{\delta-1}^{(i)}+M_{\Delta-1}^{(i)}A_\delta^{(i)}}{M_\Delta^{(i)}M_\Delta^{(i)}} \quad \text{for } \delta\neq\Delta$$

$$\Omega^{(i+1)}(x)=A^{(i)}(x)+x^{\delta-\Delta-1}\cdot M^{(i)}(x)\cdot q^{(i)}(x) \quad (5)$$
$$\sigma^{(i+1)}(x)=a^{(i)}(x)+x^{\delta-\Delta-1}\cdot m^{(i)}(x)\cdot q^{(i)}(x) \quad (6)$$

[0039] if($\deg(\Omega^{(i+1)}(x))<\Delta$)

$$A^{(i+1)}(x)=M^{(i)}(x),\ M^{(i+1)}(x)=\Omega^{(i+1)}(x)$$
$$a^{(i+1)}(x)=m^{(i)}(x),\ m^{(i+1)}(x)=\sigma^{(i+1)}(x)$$

[0040] else

$$A^{(i+1)}(x)=\Omega^{(i+1)}(x),\ M^{(i+1)}(x)=M^{(i)}(x)$$
$$a^{(i+1)}(x)=\sigma^{(i+1)}(x),\ m^{(i+1)}(x)=m^{(i)}(x)$$

[0041] else

$$\Omega(x)=\Omega^{(i)}(x),\ \sigma(x)=\sigma^{(i)}(x)\ \text{Finish}$$

[0042] where $q^{(i)}(x)=q_1{}^{(i)}x+q_0{}^{(i)}$ is the i-th dummy quotient polynomial, $\Omega^{(i+1)}(x)$ is the i-th dummy remainder polynomial, and $A_\delta{}^{(i)}$ and $M_\Delta{}^{(i)}$ are the leading coefficients of the i-th dummy dividend polynomial $A^{(i)}(x)$ with degree of δ and the i-th dummy remainder polynomial with degree of Δ, respectively. Note that if there are only errors, the erasure locator polynomial, $\Lambda(x)$ equals 1 and the Forney syndrome polynomial, T(x) should be altered to the syndrome polynomial S(x).

[0043] As compared with (3), if we assume the i-th dividend polynomial $R^{(i-2)}(x)$ to $A^{(i)}(x)$ as well as the i-th divisor polynomial $R^{(i-1)}(x)$ to $M^{(i)}(x)$, the difference in degree between $A^{(i)}(x)$ and $M^{(i)}(x)$ equaling $\delta-\Delta$ implies the decoding procedure shown above will take at most

$$\left\lceil\frac{\delta-\Delta+1}{2}\right\rceil$$

[0044] iterations to calculate the i-th remainder polynomial $R^{(i)}(x)$.

[0045] Note that our modified decoding procedure will stop at $\deg(\Omega^{(i)}(x))<\deg(\sigma^{(i)}(x))$ and in the meantime, $\sigma^{(i)}(x)$ is the errata locator polynomial $\sigma(x)$ with degree of s+v. That s and v represent the number of actual erasure(s) and error(s). Recalling $\deg(\sigma^{(0)}(x))=\deg(\Lambda(x))=s$, the degree of $\sigma^{(i)}(x)$ will increase from s to s+v. In a specific case with degree of $Q^{(i)}(x)$ in (3) all equaling one, v division operations are needed and in the decoding procedure shown above, the total number of iterations is v as a result that accomplishing each division operation takes 1 iteration with $\delta-\Delta=\deg(q^{(i)}(x))=1$. Owing to $v\leqq t$, the modified decoding procedure above requires at most t iterations for solving key equation polynomials.

[0046] The Inversionless Decoding Procedure

[0047] For eliminating the inverse operation within our modified decoding procedure, a novel inversionless decoding procedure is proposed and shown as follows:

[0048] Initial Condition:

$$\hat{A}^{(0)}(x)=x^{N-K},\ \hat{M}^{(0)}(x)=\hat{\Omega}^{(0)}(x)=T(x)$$
$$\hat{a}^{(0)}(x)=0,\ \hat{m}^{(0)}(x)=\hat{\sigma}^{(0)}(x)=\Lambda(x)$$

[0049] For (i=0 to t)

$$\delta=\deg(\hat{A}^{(i)}(x)),\ \Delta=\deg(\hat{M}^{(i)}(x))$$

[0050] if($\deg(\hat{\sigma}^{(i)}(x))\leqq\deg(\hat{\Omega}^{(i)}(x))$)

$$\hat{q}_1{}^{(i)}(x)=\hat{A}_\delta{}^{(i)}\hat{M}_\Delta{}^{(i)}$$
$$\hat{q}_0{}^{(i)}(x)=0 \text{ for } \delta\neq\Delta$$
$$\hat{q}_0{}^{(i)}(x)=\hat{M}_\Delta{}^{(i)}\hat{A}_{\delta-1}{}^{(i)}+\hat{M}_{\Delta-1}{}^{(i)}\hat{A}_\delta{}^{(i)} \text{ for } \delta=\Delta$$
$$\hat{\Omega}^{(i+1)}(x)=\hat{M}_\Delta{}^{(i)}\hat{M}_\Delta{}^{(i)}\cdot\hat{A}^{(i)}(x)+x^{\delta-\Delta-1}\cdot\hat{M}^{(i)}(x)\cdot\hat{q}^{(i)}(x) \quad (7)$$
$$\hat{\sigma}^{(i+1)}(x)=\hat{M}_\Delta{}^{(i)}\hat{M}_\Delta{}^{(i)}\cdot\hat{a}^{(i)}(x)+x^{\delta-\Delta-1}\cdot\hat{m}^{(i)}(x)\cdot\hat{q}^{(i)}(x) \quad (8)$$

[0051] if($\deg(\hat{\Omega}^{(i+1)}(x))<\Delta$)

$$\hat{A}^{(i+1)}(x)=\hat{M}^{(i)}(x),\ \hat{M}^{(i+1)}(x)=\hat{\Omega}^{(i+1)}(x)$$
$$\hat{a}^{(i+1)}(x)=\hat{m}^{(i)}(x),\ \hat{m}^{(i+1)}(x)=\hat{\sigma}^{(i+1)}(x)$$

[0052] else

$$\hat{A}^{(i+1)}(x)=\hat{\Omega}^{(i+1)}(x),\ \hat{M}^{(i+1)}(x)=\hat{M}^{(i)}(x)$$

$$\hat{a}^{(i+1)}(x)=\hat{\sigma}^{(i+1)}(x),\ \hat{m}^{(i+1)}(x)=\hat{m}^{(i)}(x)$$

[0053] else

$$\hat{\Omega}(x)=\hat{\Omega}^{(i)}(x),\ \hat{\sigma}(x)=\hat{\sigma}^{(i)}(x)\ \text{Finish}$$

[0054] where $\hat{\Omega}(x)$ and $\hat{\sigma}(x)$ are the modified errata evaluator polynomial and errata locator polynomial, respectively, It can be shown that $\hat{\sigma}(x)$ and $\hat{\Omega}(x)$ can be used to find the same error location(s) and error value(s) as the original $\sigma(x)$ and $\Omega(x)$ do. While compared with other approaches, our proposed inversionless Euclidean algorithm not only eliminates the costly inversion operation but also introduces a t-iteration decoding procedure.

[0055] The Decomposed Architecture

[0056] Here we propose a decomposed architecture from the proposed inversionless Euclidean algorithm, which works with individual coefficients of the polynomial instead of the entire polynomial as a whole.

[0057] And (7)~(8) can be decomposed as the following two equations:

$$\hat{\Omega}_j^{(i+1)}=\hat{M}_\Delta^{(i)}\hat{M}_\Delta^{(i)}\cdot\hat{A}_j^{(i)}+\hat{M}_{j-(\delta-\Delta-1)}^{(i)}\cdot\hat{q}_0^{(i)}+\hat{M}_{j-(\delta-\Delta)}^{(i)}\cdot\hat{q}_1^{(i)}\ 0\le j\le\delta-2 \quad (9)$$

$$\hat{\sigma}_k^{(i+1)}=\hat{M}_\Delta^{(i)}\hat{M}_\Delta^{(i)}\cdot\hat{a}_k^{(i)}+\hat{m}_{k-(\delta-\Delta-1)}^{(i)}\cdot\hat{q}_0^{(i)}+\hat{m}_{k-(\delta-\Delta)}^{(i)}\cdot\hat{q}_1^{(i)}\ 0\le k\le\phi \quad (10)$$

[0058] where δ and Δ represent the degree of $\hat{A}^{(i)}(x)$ and $\hat{M}^{(i)}(x)$ respectively, and $\hat{\Omega}_j^{(i+1)}$ and $\hat{\sigma}_k^{(i+1)}$ corresponds to the j-th and k-th coefficient of $\hat{\Omega}^{(i+1)}(x)$ with degree of $\delta-2$ and $\hat{\sigma}^{(i+1)}(x)$ with degree of ϕ at the i-th iteration. From (9)~(10), if $\hat{M}_\Delta^{(i)}\hat{M}_\Delta^{(i)}$, $\hat{q}_0^{(i)}$ and $\hat{q}_1^{(i)}$ can be calculated in advance, there only three finite-field multiplications needed to compute $\hat{\Omega}_j^{(i+1)}$ and $\hat{\sigma}_k^{(i+1)}$. The detailed cycle operation of our inversionless decomposed architecture can be seen in Table 2. For simplifying notations, we let $\delta-\Delta=1$ without loss of generality.

TABLE 2

Cycle	$\hat{\Omega}^{(i+1)}(x)$ and $\hat{\sigma}^{(i+1)}(x)$		
Initialization	$w=\hat{M}_\Delta^{(i)}\hat{M}_\Delta^{(i)}$		
	$\hat{q}_0^{(i)}=\hat{M}_\Delta^{(i)}\hat{A}_{\delta-1}^{(i-1)}+$	$\hat{M}_{\Delta-1}^{(i)}\hat{A}_\delta^{(i-1)}$	
j = 0	$\hat{q}_1^{(i)}=\hat{M}_\Delta^{(i)}\hat{A}_\delta^{(i-1)}$		
	$\hat{\Omega}_0^{(i+1)}=w\cdot\hat{A}_0^{(i-1)}+$	$\hat{M}_0^{(i)}\cdot\hat{q}_0^{(i)}$	
j=1	$\hat{\Omega}_1^{(i+1)}=w\cdot\hat{A}_1^{(i-1)}+$	$\hat{M}_1^{(i)}\cdot\hat{q}_0^{(i)}+$	$\hat{M}_0^{(i)}\cdot\hat{q}_1^{(i)}$
...	...	...	...
j = δ − 2	$\hat{\Omega}_{\delta-2}^{(i+1)}=w\cdot\hat{A}_{\delta-2}^{(i-1)}+$	$\hat{M}_{\Delta-1}^{(i)}\cdot\hat{q}_0^{(i)}+$	$\hat{M}_{\Delta-2}^{(i)}\cdot\hat{q}_1^{(i)}$
j = δ − 1	$\hat{\Omega}_{\delta-1}^{(i+1)}=w\cdot\hat{A}_{\delta-1}^{(i-1)}+$	$\hat{M}_\Delta^{(i)}\cdot\hat{q}_0^{(i)}+$	$\hat{M}_{\Delta-1}^{(i)}\cdot\hat{q}_1^{(i)}=0$
j = δ	$\hat{\Omega}_\delta^{(i+1)}=w\cdot\hat{A}_\delta^{(i-1)}$	+	$\hat{M}_\Delta^{(i)}\cdot\hat{q}_1^{(i)}=0$
k = 0	$\hat{\sigma}_0^{(i+1)}=w\cdot\hat{a}_0^{(i)}+$	$\hat{m}_0^{(i)}\cdot\hat{q}_0^{(i)}$	
k = 1	$\hat{\sigma}_1^{(i+1)}=w\cdot\hat{a}_1^{(i)}+$	$\hat{m}_1^{(i)}\cdot\hat{q}_0^{(i)}+$	$\hat{m}_0^{(i)}\cdot\hat{q}_1^{(i)}$
...	...	...	...
k=ψ−1	$\hat{\sigma}_{\psi-1}^{(i+1)}=w\cdot\hat{a}_{\psi-1}^{(i)}$	$\hat{m}_{\psi-1}^{(i)}\cdot\hat{q}_0^{(i)}+$	$\hat{m}_{\psi-2}^{(i)}\cdot\hat{q}_1^{(i)}$
k=ψ	$\hat{\sigma}_\psi^{(i+1)}=w\cdot\hat{a}_\psi^{(i)}$	$\hat{m}_\psi^{(i)}\cdot\hat{q}_0^{(i)}$	$\hat{m}_{\psi-1}^{(i)}\cdot\hat{q}_1^{(i)}$

[0059] It is evident from Table 1 that, at cycle j=0, the computation of $\hat{\Omega}_0^{(i+1)}$ requires $w=\hat{M}_\Delta^{(i)}\hat{M}_\Delta^{(i)}$ and $\hat{q}_0^{(i)}$, which have been calculated at the initialization cycle. Similarly, at cycle $j\ge1$, the computation of $\hat{\Omega}_j^{(i+1)}$ also requires $\hat{q}_1^{(i)}$, which has been calculated at cycle j=0. Note that each cycle needs three finite-field multiplications and the calculations process of $\hat{\sigma}^{(i+1)}(x)$ is similar to that of $\hat{\Omega}^{(i+1)}(x)$.

[0060] The inversionless decomposed Euclidean algorithm shown above suggests a 3-FFM implementation of the key equation solver, which is illustrated in **FIG. 2**. The branch labeling in **FIG. 2** corresponds to a particular time instance. As compared with Table 1, **FIG. 2**(*a*) shows the initialization cycle, when j=0, refer to Table 1, is $w=\hat{M}_\Delta^{(i)}\hat{M}^{\Delta(i)}$ is computed by a finite field multiplier **34**, equation $\hat{q}_0^{(i)}(x)=\hat{M}_\Delta^{(i)}\hat{A}_{\delta-1}^{(i-1)}+\hat{M}_{\Delta-1}^{(i)}\hat{A}_\delta^{(i-1)}$, wherein $\hat{M}^{\Delta(i)}\hat{A}_{\delta-1}^{(i-1)}$ is computed by a finite field multiplier **30**, $\hat{M}^{\Delta-1(i)}\hat{A}_\delta^{(i-1)}$ is computed by a finite field multiplier **32**, these two terms are added by a finite field adder **36** to have $\hat{q}_0^{(i)}$, as shown in **FIG. 2***a*. **FIG. 2**(*b*) indicates the calculation cycle for $\hat{q}_1^{(i)}$ and $\hat{\Omega}_0^{(i+1)}$. Since $\hat{q}_1^{(i)}=\hat{M}_\Delta^{(i)}\hat{A}_\delta^{(i-1)}$, the finite field multiplier **40** is used to compute $\hat{q}_1^{(i)}$, since $\hat{\Omega}_0^{(i+1)}=w\cdot\hat{A}_0^{(i-1)}+\hat{M}_0^{(i)}\cdot\hat{q}_0^{(i)}$, the finite field multiplier **44** realize $w\cdot\hat{A}_0^{(i-1)}$ and the finite field multiplier **42** realize $\hat{M}_0^{(i)}\cdot\hat{q}_0^{(i)}$, these two terms are added by a finite field adder **36** to have $\hat{\Omega}_0^{(i+1)}$.

[0061] The process for computing other coefficients of $\hat{\Omega}^{(i+1)}(x)$ is expressed in **FIG. 2**(*c*), when $j\ge1$, refer to Table 1, $\hat{\Omega}^{(i+1)}(x)$ can be obtained by finite field multiplier **50**,**52**, **54** and a finite field adder **56**. Because the computation process of $\hat{\sigma}^{(i+1)}(x)$ is similar to that of $\hat{\Omega}^{(i+1)}(x)$, the hardware used to compute $\hat{\Omega}^{(i+1)}(x)$ can be reconfigured to calculated $\hat{\sigma}^{(i+1)}(x)$, which is presented in **FIG. 2**(*d*), the hardware is similar to **FIG. 2***c*, with multipliers **60**,**62**,**64** and an adder **66** to compute $\hat{\sigma}^{(i+1)}(x)$ for k=1 to ψ, as illustrate in Table 1.

[0062] This architecture can be used for error-only correction as well as error-and-erasure correction. Compared to existing proposals requiring 6t to 8t FFMs, the preferred embodiment of the present invention significantly reduces hardware complexity down to 3 FFMs. However, in order to finish the i-th iteration, the architecture of the preferred embodiment requires $\delta+\psi+1$ cycles whereas prior art architectures requires only two to three cycles. The additional time required for generating the data under the architecture of the present invention does not slow down the overall system processing speed. Due to the overall system processing speed dominated by the syndrome calculator and Chien Search, each taking N cycles to finish, our architecture slowing down the Euclidean algorithm (till taking N cycles) will not impact the decoding speed.

[0063] Additionally, the method and apparatus of the present invention also minimize the amount of required registers. Recalling (9) and (10), ψ representing the degree of $\hat{\sigma}^{(i+1)}(x)$ is equivalent to the degree of $\mu^{(i)}(x)$ in (4) and similarly, Δ representing the degree of $\hat{M}^{(i)}(x)$ is equivalent to that of $R^{(i-1)}(x)$ in (3). As shown earlier, $\deg(\mu^{(i)}(x))+\deg(R^{(i-1)}(x))=N-K+s\le2t+\rho$, where t and ρ represent the number of errors and erasures in the decoding of codewords and consequently, in the preferred embodiment of the present invention, $2t+\rho+2$ registers are used to store the coefficients of $\hat{M}^{(i)}(x)$ and $\hat{\sigma}^{(i+1)}(x)$, and another $2t+\rho$ registers can be used for storing the coefficients of $\hat{m}^{(i)}(x)$ and $\hat{\Omega}^{(i+1)}(x)$. Hence, calculating $\hat{\Omega}^{(i+1)}(x)$ and $\hat{\sigma}^{(i+1)}(x)$ iteratively totally requires $4t+2\rho+2$ registers and if there are only errors corrected, the amount of required registers is 4t+2 and the previously proposed architectures requiring 6t to 8t registers.

[0064] Furthermore, the preferred embodiment of the present invention can also be used to calculate the Forney syndrome polynomial, T(x), which is defined as:

$$T(x)=S(x)\Lambda(x) \bmod x^{N-K} \quad (11)$$

[0065] where

$$\Lambda(x)=\prod_{j=1}^{s}(1+\chi_j x)$$

[0066] is the erasure locator polynomial and χ_j is the j-th erasure magnitude. T(x) can be obtained by following procedures:

[0067] Initial Condition

$$T^{(0)}(x)=S(x)$$

[0068] For(i=0 to t)

[0069] if(2i<s)

$$\Lambda^{(i)}(x)=(1+\chi_{2i}x)(1+\chi_{2i+1}x) \quad (12)$$

$$T^{(i+1)}(x)=T^{(i)}(x)\cdot\Lambda^{(i)}(x) \bmod X^{N-K} \quad (13)$$

[0070] else

$$T(x)=T^{(i)}(x) \text{ Finish}$$

[0071] where $\Lambda^{(i)}(x)$ is the i-th auxiliary polynomial for computing the i-th iteration Fonrey syndrome polynomial, $T^{(i+1)}(x)$. Note that $\Lambda^{(i)}(x)$ can be expressed as $1+\Lambda^{1(i)}x+\Lambda_2{}^{(i)}x^2$ and $T^{(i+1)}(x)$ can be decomposed as the following results:

$$T_\tau{}^{(i+1)}=T_\tau{}^{(i)}+T_{\tau-1}{}^{(i)}\cdot\Lambda_1{}^{(i)}+T_{\tau-2}{}^{(i)}\cdot\Lambda_2{}^{(i)}\ 0\leqq\tau\leqq N-K-1 \quad (14)$$

[0072] It is evident that the process calculating the τ-th coefficient, $T_\tau{}^{(i+1)}$, is very similar to (9) or (10), and therefore, the 3-FFM architecture can be used to obtain the Forney syndrome polynomial T(x), which is illustrated in **FIG. 2**(*e*), as refer to equation (14), the second term is computed by a finite field multiplier**72**, the third term is computed by a finite field multiplier**70** and these two terms and the first term are added by a finite field adder **74** to have $T_\tau{}^{(i+1)}$.

[0073] Application Conditions

[0074] The total number of cycles required to compute $\hat{\sigma}(x)$ and $\hat{\Omega}(x)$ using the 3-FFM architecture of the preferred embodiment is of interest in considering the potential impact on the overall system performance. From the proposed iterative decoding process, $0\leqq j\leqq\delta-2$ in (9) and $0\leqq\lambda\leqq\psi$ in (10) implying the number of cycles required to compute $\hat{\Omega}^{(i+1)}(x)$ is δ−1 and calculating $\hat{\sigma}^{(i+1)}(x)$ needs ψ+1 cycles in the i-th iteration. However, one more cycles is needed to get $\hat{q}_1{}^{(i)}$ and $\hat{q}_0{}^{(i)}$, and the proposed decoding procedure requires δ+ψ+1 cycles in one iteration totally. From ψ+Δ=N-K+s and $\delta-\Delta\leqq 1$, it is clear that $\delta+\psi+1\leqq N-K+s+2\leqq 2t+\rho+2$. For RS (N,K) code for correcting t errors and ρ erasures, the total number of cycles required in our t-iteration decomposed inversionless architecture is less than $2t^2+\rho t+2t$. Table 3 shows the maximum number of cycles for different RS (N,K) codes with N-K ranging from 4 to 16. If N is larger than the number of cycles required, then our 3-FFMs architecture can be applied to reduce the hardware complexity while maintaining the overall decoding speed.

TABLE 3

N − K	t	ρ	cycles	t	ρ	Cycles
4	2	—	12	1	2	6
6	3	—	24	2	2	16
8	4	—	40	3	2	30
10	5	—	60	4	2	48
12	6	—	84	5	2	70
14	7	—	112	6	2	96
16	8	—	144	7	2	126

[0075] There are many applications of BCH and RS codes in communications and storage systems that benefit from methods and apparatus of the present invention. For example, Digital Versatile Disks (DVDs) use a RS product code which is **(182,172)** in the row direction and **(208,192)** in the column direction. Digital TV broadcasting uses a **(204,188)** RS code. CD-ROM uses a number of smaller **(32,28)** and **(28,24)** RS codes. In the optical fiber submarine cable systems, RS **(255,239)** code is used and standardized to provide burst error correcting capability. In wireless communications, the AMPS cellular phone system uses **(40,28)** and **(48,36)** binary BCH codes, which are shortened codes of the **(63,51)** code. The **(63,51)** code, which can correct up to 2 errors (N-K=12,m=6), requires fewer than 12 cycles (t=2, row 1 of Table 3). All of these applications, as well as many others, can benefit from the method and apparatus of the present invention.

[0076] Although the present invention has been described in terms of specific embodiments, it is anticipated that alterations and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modifications as fall within the true spirit and scope of the invention.

What is claimed is:

1. An apparatus for solving key equation polynomials in decoding error correction codes, a novel inversionless decomposed architecture which is frequently used in BCH and Reed-Solomon decoders comprising:

a syndrome calculator that received codewords and output a syndrome polynomial to a key equation solver;

a key equation solver that calculated error locator polynomial and error evaluator polynomial and output error location;

a Chein Search that received said error locator polynomial and input a result to an error value calculator and output said error location;

an error value calculator that received signal from said key equation solver and Chein Search, output an error value.

2. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus is used for BCH and Reed-Solomon (RS) decoders.

3. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus is applied to BCH and Reed-Solomon (RS) decoders which is a kind of inversionless decomposed architecture.

4. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus can be applied to the correction of errors as well as erasures.

5. An apparatus for solving key equation polynomials in decoding error correction codes in claim 1, wherein said method and apparatus is applied in inversionless Euclidean.

6. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus can eliminate the finite-field inverter (FFI) to finish.

7. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus is only needed t iteration decoding procedure.

8. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus including said decomposed technique which can also drastically reduce the required number of finite-field multipliers (FFMs) from 4t~6t to 3.

9. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus including said decomposed technique that uses only $4t+2\rho+4$ registers.

10. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus including said decomposed technique that no FFIs is presented to implement the inversionless Euclidean algorithm.

11. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus can use to calculate the Forney syndrome polynomial.

12. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said apparatus is further operable in communication.

13. A method for solving key equation polynomials in decoding error correction codes. In particular, a novel method for inversionless decomposed architecture which is frequently used in BCH and Reed-Solomon decoders executable instructions for:

(a) received said codewords and calculate said syndrome;

(b) produced said errata locator polynomial and errata evaluator polynomial;

(c) search said error location;

(d) calculated said error value.

14. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method is used for BCH and Reed-Solomon (RS) decoders.

15. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method is applied to BCH and Reed-Solomon (RS) decoders which is a kind of inversionless decomposed architecture.

16. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method can be applied to the correction of errors as well as erasures.

17. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method is applied in inversionless Euclidean.

18. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method can eliminate the finite-field inverter (FFI) to finish.

19. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method is only needed t iteration decoding procedure.

20. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method including said decomposed technique which can also drastically reduce the required number of finite-field multipliers (FFMs) from 4t~6t to 3.

21. An apparatus for solving key equation polynomials in decoding error correction codes according to claim 1, wherein said method including said decomposed technique that uses only $4t+2\rho+4$ registers.

22. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method including said decomposed technique which no FFIs is presented to implement the inversionless Euclidean algorithm.

23. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method including said decomposed technique which can also use to calculate the Forney syndrome polynomial.

24. A method for solving key equation polynomials in decoding error correction codes according to claim 13, wherein said method and apparatus is further operable in communication.

25. A method for solving key equation polynomials in decoding error correction codes. In particular, a novel method for inversionless decomposed architecture which is frequently used in BCH and Reed-Solomon decoders, wherein improving process including;

(a) improved the speed of said Educlidean algorithm;

(b) embellished said decoded procedure to reduce half decoded result;

(c) combined the calculate which said errata locator polynomial and errata evaluator polynomial.

26. A mothed as recited in claim 25, wherein said Educlidean algorithm and time is shared said finite-field multipliers (FFMs).

27. A mothed as recited in claim 25, wherein said method can reduce said hardware area.

28. A mothed as recited in claim 25, wherein said modified Educlidean algoridean is a decomposed architecture, eliminated the limit of finite-field inversionless

29. A mothed as recited in claim 25, wherein said inversionless Educlidean algoridean including total iteration number of degree is less than t but also other architectures requires at most 2t interations.

30. A mothed as recited in claim 28, wherein said inversionless Educlidean algoridean use the degree of said error locator polynomial increase from $\rho+1$ to $\rho+t$.

31. A mothed as recited in claim 25, wherein said inversionless Educlidean algoridean, the number of total iterations in our modified procedure is less than t.

32. A method for solving key equation polynomials in decoding error correction codes. In particular, a novel method for inversionless decomposed architecture which is frequency used in BCH and Reed-Solomon decoders including:

(a) each iteration could eliminate at least one degree;

(b) combined the hardware of said errata locator polynomial and errata evaluator polynomial;

(c) a number of FFMs is reduced to 3.

33. A mothed as recited in claim 32, wherein said speed of inversionless Educlidean algoridean slowing down, but it will not impact the decoding speed.

34. A mothed as recited in claim 32, wherein said BCH and Reed-Solomon (RS) decoder, Digital Versatile Disks (DVDs) use a RS product code which is **(182,172)** in the row direction and **(208,192)** in the column direction.

35. A mothed as recited in claim 32, wherein said BCH and Reed-Solomon (RS) decoder, digital TV broadcasting uses a **(204,188)** RS code.

36. A mothed as recited in claim 32, wherein said BCH and Reed-Solomon (RS) decoder, CD-ROM uses a number of smaller RS codes, including **(32,28),(28,24)**.

37. A mothed as recited in claim 32, wherein said BCH and Reed-Solomon (RS) decoder, in wireless communications, the AMPS cellular phone system uses **(40,28)** and **(48,36)** binary BCH codes, which are shortened codes of the **(63,51)** code.

* * * * *