International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 8, August 2017)

High computational Speed with an Optimal Utilization of Resources for Forward Error Correcting Decoder

Mahender Veshala[1], T. Srinivasulu[2]

[1,2]*Dept. of E.C.E, Kakatiya University College of Engineering and Technology, Warangal, T.S. India*

***Abstract*—Presently the Decoding can be done by using Chien search Algorithm with Bose–Chaudhuri–Hochquenghem (BCH) codes in receivers. In general chien search block involves massive computational complexity that occupies more area of the total decoding logic. This decoding process becomes more complex for correcting errors in parallel with more hardware. In this paper, proposed method is high speed with an optimal utilization of resources for forward error correcting Chien search decoder or BCH Decoder and also performs the Chien search error correcting process in parallel basis with reduced hardware. In general shift operations require less hardware than multiplication. Hence Cumulative Glois Finite Field Multiplier (CGFM) replaced with Shift operation.**

***Index Terms*— Computational speed, Cumulative Galois Finite field Multiplier, Hardware requirement, Optimal utilization.**

I. Introduction

The Bose–Chaudhuri–Hochquenghem (BCH) codes are a large class of powerful random error-correcting cyclic codes, which have received much attention from academia and industry since their invention in 1959. They have been widely used in communications and data storage systems, including satellite communications, cellular networks, CD Rom, Mass Storage Systems, wireless broadband, etc. require high decoding throughput as well as a large error correcting capability [3]–[6]. In order to satisfy these competing demands, parallel implementation is needed. Since the parallel implementation requires complicated hardware, the design of the area-efficient decoder.

This paper focuses on an optimal implementation of the Chien search process that requires less area with high speed in the decoding process of BCH codes [7]. There are some literatures that suggest the efficient Chien search algorithms for low error correcting capability [8], [9]. It will present an alternative method for implementing the Chien search, which shows better performance especially for a high error correcting capability with high speed. The table look-up method that is described in plenty of CRC-related literatures is adopted in the Chien search process in order to parallelize the Galois field arithmetic functions [3], [4], [10].

The rest of this paper is organized as follows. Section II briefly explains the general BCH decoder Architecture and its function.

Section III introduces the conventional Chien search architectures for decoding purpose. In Section IV the parallelization method used in the Galois field arithmetic described and proposed an optimal utilization of resources for forward error correcting Chien Search decoder architecture, and estimate the approximate area of the proposed architecture. The experimental results for more accurate area estimation are shown in Section V. Finally, the concluding remarks are given in Section VI.

II. Convolution BCH Decoder Architecture

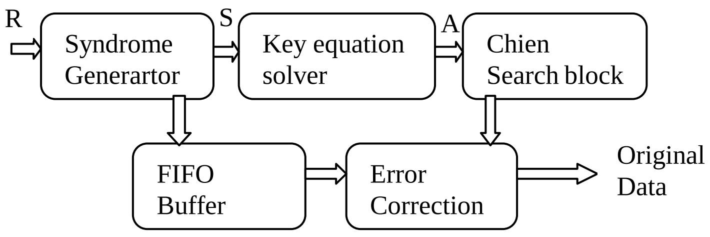


Figure 1. BCH Decoder Architecture

The decoding of binary BCH codes consists of the following three steps: a) at first, the received polynomial R(x) is fed to a syndrome computation block, and generates syndrome polynomial S(x); b) then the key equation S(x).A(x) = Ω(x) $|x^{2t}|$ is solved in the key-equation solver block; c) eventually; the errors are corrected by finding out the roots of error locator polynomials using the Chien search algorithm. Among the three decoding blocks, the Chien search block involves a massive computational complexity that occupies more than 75% area of the total decoding logic in forward error correction devices [7], [8]. From Fig. 1. an original Data is nothing but actual encoded data from transmitter. FIFO buffer received noisy polynomial and it gives to error correction block, which can be implemented with logical XOR operation.

III. Conventional Chien Search Architecture

We denote standard polynomial (over the finite field GF (2^t)) [4] for BCH Codes whose roots wish to determine as:

$$A(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \dots + \lambda_t x^t \qquad (1)$$

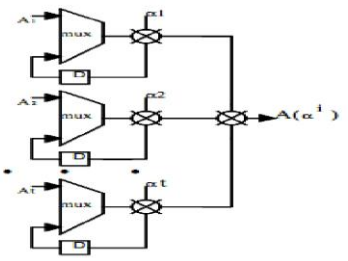


Figure 2. Serial Chien Search.

Where ‘ λ_j’ is error locator polynomial coefficients and j = 0, 1,2,………t. Conceptually, evaluate A (β) for each non-zero $\beta=\alpha^i$ in GF (2^t) is equal to ‘0’. The corresponding ‘β’ becomes one of the root for the above equ.1. By considering an (n, k, t) binary BCH code which has the codeword length n= 2^P -1 for some positive integer ‘p’ the information length k = n - pt, and the error correcting capability of t-bits. In the decoding process, the error locator polynomial A(x) can be obtained by performing syndrome calculation and by solving the key equation with key-equation solver block. Upon receiving the error locator polynomial A(x), the Chien search block exhaustively examines that ‘$\alpha^{i'}$ is a root of A(x) for i=0,1,2,3,……….n-1; i.e., it verifies if the following equation yields zero or not:

$$A(\alpha^i)=\sum_{j=0}^{t}A_j\alpha^{ij}=\sum_{j=1}^{t}A_j\alpha^{ij}+1 \qquad (2)$$

The serial implementation of the Chien search block is straight-forward from the upper equation. The conventional Chien search circuit is illustrated in Fig. 2. It produces an error vector ‘e’ in such a way that, if α^i is a root, then the (n-i)th component $e_{(n-i)}$=1; otherwise $e_{(n-i)}$=0 for all $0 \le i \le n-1$.

IV. The Parallelization Method in Galois Field Arithemetic for the Chien Search Process

Galois field (2^m) concept plays important role in finding the roots for error location from error polynomial. In this concept all modulo additions are performed with logical XOR operations. Let ‘m’ be the order of error polynomial and it has ability to correct the ‘m’ errors. Error correction can be decided by an order of the error locator polynomial. This process involved trial and error and iteration

Let $A(x)=A_mx^m+A_{m-1}x^{m-1}\ldots+\ldots\ldots+A_1x+A_o$, mth order error locator polynomial, in which store all the possible roots of this equation in look-up table , such that it provides the exactly error location .This would be done with the help of Galois field properties.

Let ‘α’ be a root of f(x). Since A (α) =0, then it gives,

$$\alpha^m = A_{m-1}\alpha^{m-1} + A_{m-2}\alpha^{m-2}+\ldots\ldots.+A_1\,\alpha+A_0 \qquad (3)$$

Put $x=\alpha^i$ in A(x), where ‘i’ represents location of error from (m-1)th to 0. For example if calculate the error in 1st position then ‘i=1’ is considered as α in A(x), then it gives,

$$A\ (\alpha) = (A_m\alpha^{\ m} + A_{m-1}\ \alpha^{\ m-1}+\ldots +A_1\ \alpha\ +A_o) \qquad (4)$$

If equate the eq. (4) =0, define all factors associated with α in the look-up table based on irreducible polynomial of the error locator polynomial with Galois field properties. Similarly continuing this process until codeword completed.

The process is done in serially, it takes number of clock cycles required are more, but hardware requirement is less. Hence in order to increase the processing speed, it has to be taken the process in parallel manner, which is parallel connection of Fig.2.

In which all possible number of ways covered to determine the error location. For example our codeword is 8-bit length,8 clock cycles are required for serial process, but in parallel process which are reduced to 1.approximately 91% of clock cycles are saved, so that specifies enhancement in speed. These all values are stored in ‘m’ bit registers for every process calculation, let us first assume that the arbitrary element A is stored in the m-bit register with the vector form $[a_{m-1}, a_{m-2}, \bullet\bullet\bullet, a_0]$. Then the CGFFM of Eq. 2 can be accomplished in Galois field with the procedure depicted with parallel connections with shift Fig. 1; i.e., it can be realized by a shift-left operation of A by 1-bit followed by an addition of α^m according to the overflowed value. The process can be evaluated by using the following table I for the following irreducible polynomial equation E(z):

$$E(z) = A_3z^3 +A_1z+A_o \qquad (5)$$

Table.I.
Galois Field Calculation for α th roots.

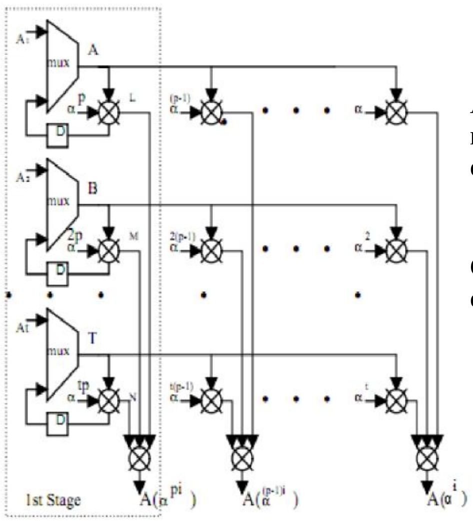
Exponent	**Polynomial**	**Binary**
0	0	000
$\boldsymbol{\alpha^0}$	1	001
$\boldsymbol{\alpha^1}$	z	010
$\boldsymbol{\alpha^2}$	z^2	100
$\boldsymbol{\alpha^3}$	z+1	011
$\boldsymbol{\alpha^4}$	z^2+z	110
$\boldsymbol{\alpha^5}$	z^2+z+1	111
$\boldsymbol{\alpha^6}$	z^2+1	101
$\boldsymbol{\alpha^7=\alpha^0}$	1	001
$\boldsymbol{\alpha^8=\alpha^1}$	z	010

The binary equivalent ($\alpha^2\alpha^1\alpha^0$) determined by using the irreducible polynomial. If calculate α^3, then eq.(5)=0.i.e., α^3= α^2+1.because of modulo addition can be done with XOR operation, which is one of the property of Galois field.

The above roots store in the lookup table for evaluating the error locator polynomial, according to the Galois field properties, where ‘z’ is the polynomial parameter.

In parallel method of evaluation can be done by eliminating the constant ‘1’ in the equation (2).

$$A(\alpha^i) = \sum_{j=1}^{t} A_J \alpha^{ij} = \sum_{j=1}^{t} A_J \alpha^{ij} \qquad (6)$$



A,B,.. multiplexer outputs

L,M,… CGFFM outputs.

Figure 3. Parallel Chien Search

In the proposed method, once multiplication completed, CFFA receives the m-bit results to ‘t’ CFFMs, then adds them up such that

$$A(\alpha^i) = \sum_{j=1}^{t} (A_J \alpha^{ij}) R_{f(x)} \qquad (7)$$

$$A(\alpha^i) = R_{f(x)}[\alpha^i A_1 + \alpha^{2i} A_2 + ... + \alpha^{it} A_t] \qquad (8)$$

$$A(\alpha^i) = R_{f(x)} a(x) \qquad (9)$$

Where $R_{f(x)} = \dfrac{a(x)}{f(x)}$; Here $a(x)$ varies with the error location, if consider LHS position error checking then

$$a(x) = \alpha^0 A_1 + \alpha^{2*0} A_2 + ... + \alpha^{0*t} A_t .$$
$$a(x) = A_1 + A_2 + ... + A_t . \qquad (10)$$

$f(x)$ Is an irreducible polynomial as consider earlier.

Similarly if observe the error location in the 2nd position from LHS then substitute α^1 in place of α^0 in the previous iteration, that is

$$A(\alpha^1) = R_{f(x)}[\alpha^1 A_1 + \alpha^2 A_2 + ... + \alpha^{1t} A_t]. \qquad (11)$$

$$a(x) = \alpha^1 A_1 + \alpha^{2*1} A_2 + ... + \alpha^{1*t} A_t \qquad (12)$$

Likewise the optimization completed. The process of completion depends on the order of the error locator polynomial that is if mth order error locator polynomial, and then the iteration continuing up to the corresponding m roots of the polynomial.

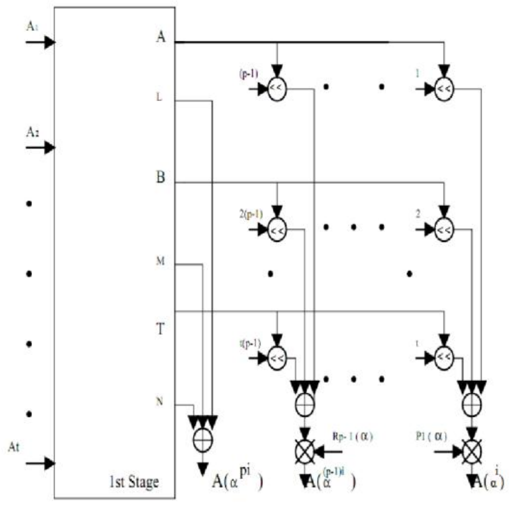


Figure 4. Proposed Method

In the conventional Chien search block CFFMS are replaced by the some number of shifters. These shifters consume less cell area, because it represents a wire in the hardware design. If, consider both irreducible and error locator polynomials are identical then architecture requirement associated with the process same. Instead of designing different hardware go for similar hardware. Hence this can say that hardware reducible scheme.

V. Experimental Results

The following results can determine by considering an irreducible polynomial eq. (5) and the error locator polynomial equation.

$$A(x) = A_2 x^2 + A_1 x + A_o \qquad (13)$$

Parallel Chien search Method:

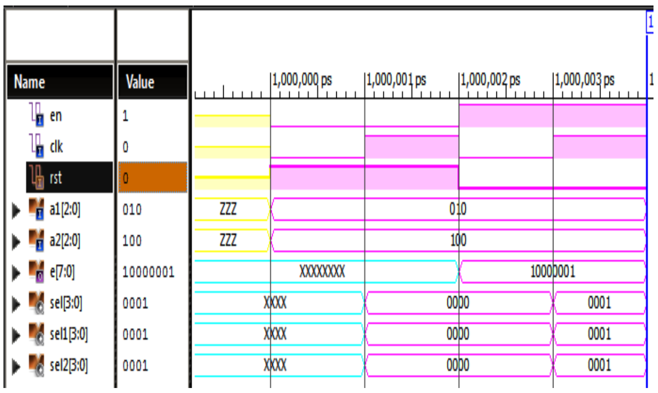


Figure 5. Parallel Chien Search

This simulation shows that error calculation in parallel manner. This requires 1 clock cycle to locate the error vector of 8 bits. Below table II shows that hardware requirement for the parallel implementation of Chien search.

Table.II.
Parallel Chien Search

Device Utilization summary	(parallel method)		
Logic Utilization	Used	Available	Utilization
Number of Slices	8	960	0%
Number of slice Flip-Flops	4	1920	0%
Number of 4 input LUTs	15	1920	0%
Number of bonded IOBs	17	66	25%
Number of GCLKs	1	24	4%

Proposed method:

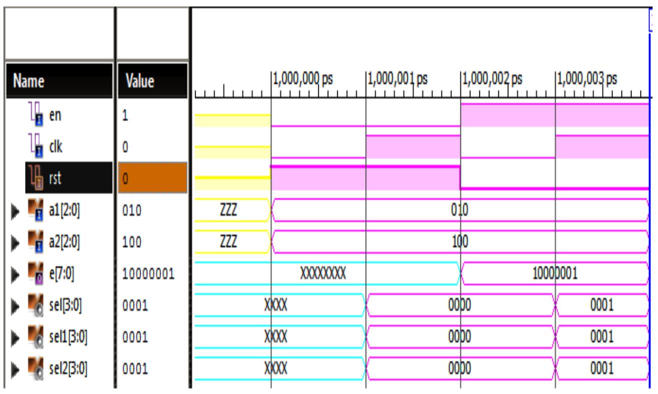


Figure 6. Proposed method

Table.III.
Proposed method utilization

Device Utilization summary	(proposed method)		
Logic Utilization	Used	Available	Utilization
Number of Slices	3	960	0%
Number of slice Flip-Flops	4	1920	0%
Number of 4 input LUTs	6	1920	0%
Number of bonded IOBs	11	66	16%
Number of GCLKs	1	24	4%

This simulation shows the parallel implementation of Chien search algorithm by using shift operations. The experimental result shows that hardware utilization for the Chien search by using shift operations is less when compared to conventional Chien search parallel method.

VI. CONCLUSION

It presents an optimal utilization of resources for efficient architecture with parallel implementation of the Chien search process in BCH decoder for forward correcting codes. By using the Galois field properties determined the roots of an error locator polynomial with reducible memory utilization.

The proposed method provides arithmetic XOR operations into normal operations, which are then converted to simple shift operations by changing the order of calculations. Since the number of finite field multiplications is much reduced, this method yields much better results when compared to previous optimizations that lower the complexities of finite field multipliers by redundant factor elimination.

The transformation also lowered the complexity of finite field adders and shift operations do not require any adder cells. The effectiveness of the proposed method is validated by accurate estimation on various error correcting capabilities and parallel factors.

REFERENCES

[1] E. R. Berlekamp, "Algebraic coding theory," McGraw-Hill, New York, 1968.
[2] E. R. Berlekamp, "Bit-serial Reed-Solomon encoders," IEEE Trans. Information Theory, vol. 28. pp. 120-126, Nov. 2008.
[3] T. Beth, D. Gollmann, "Algorithm engineering for public key algorithms," IEEE Trans. on Selected Areas in Communications. vol. 7, pp. 458-465. May 2006.
[4] R. E. Blahut, "Theory and practice of error-control codes," Addison-Wesley, Reading, MA, 2007.

[5] R. C. Bose, D.K. Ray-Chaudhuri, “On a class of error correcting binary group Codes,” Inf. Control, 3, pp. 68-79, March 2009.

[6] R. K. Brayton, G.H. Hachtel, A.L. Sangiovanni-Vincentelli, “Multilevel logic synthesis,” Processing on the IEEE, vol. 78, no. 2, Feb. 2005.

[7] H. O. Burton, “Inversionless decoding of binary BCH code,” IEEE Trans., 2005, IT-17, (4), pp. 464-466.

[8] J. Cho and W. Sung, “Strength reduced parallel Chien Search Architecture for Strong BCH codes,” IEEE Trans. on Circuits and Systems, vol. 55.pp. 427-431, May 2008.

[9] C. S. Chen, Y. W. Tsay, T. T. Hwang, A. C. H. Wu, Y. L. Lin, “Combining technology mapping and placement for delay-minimisation in FPGA design,” IEEE Transaction in Computer-Aided Design of Integrated Circuit and Systems, vol. 14, no. 9, Sep. 2007.

[10] R. T. Chien, “Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes,” IEEE Trans. Inf. Theory, IT10, pp. 357-363, October 2008.

[11] M. Damiani, J. C. Y. Yang, G. D. Micheli, “Optimisation of combinational logic circuit based on compatible gates,” IEEE Transaction in Computer-Aided Design of Integrated Circuit and Systems, vol. 14, no. 11, Nov. 2006.

[12] H. M. Deitel, P. J. Deitel, “C: How to program,” Prentice-Hall, New Jersey, 1992.

[13] M. D. Edwards, “Automatic logic synthesis techniques for digital systems,” Macmillan, 1992.

[14] Exemplar Logic Inc. “Galileo, User manual,” Alameda, 1995.

[15] C.Paar, “Optimized arithmetic for Reed-Solomon encoders,” in proc.IEEE Int.Inf.Thoery, Uim, Germani, pp. 250-253, Jun-Jul. 1997.

[16] Hanho Lee, “High-Speed VLSI Architecture for parallel Reed-Solomon Decoder,” IEEE Trasactions on very large scale Integration (VLSI) Systems, vol.11, pp.288-294, April 2003.

[17] L. Song and K. K. Parthi, “Low complexity modified Mastrovito multipliers over finite fields GF(2^m),” in proc. of IEEE ISCAS, vol.1, Orland, FL, pp.508-512, May 1999.