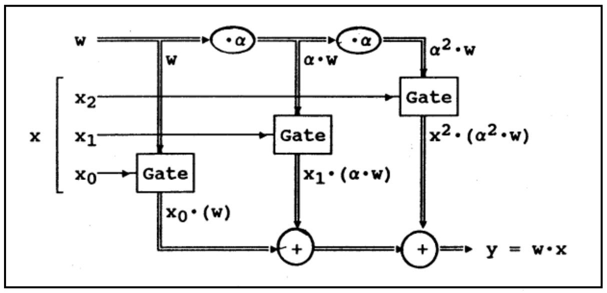
Extending the Life of Software BCH Encoders and Decoders

For Flash Memory Applications



A ChannelScience White Paper

Written by

Neal Glover

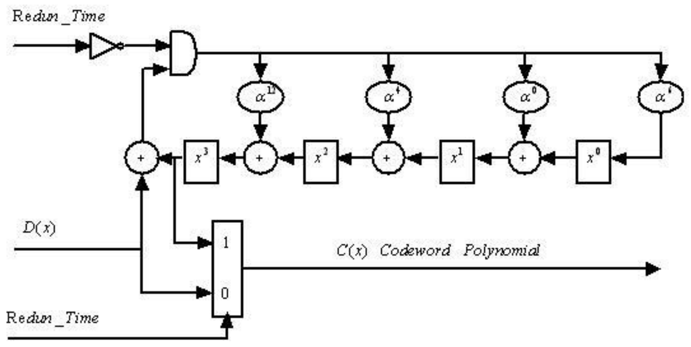
August 9, 2010

ChannelScience
Detecting the Future of Data Storage SM

7300 Cody Court
Plano TX 75024-3837 USA
connect@ChannelScience.com

(972) 814-3441 Voice
(972) 208-9095 FAX

Table of Contents



1. Executive Summary

It may be possible to extend the life of software and hybrid BCH systems for Flash memory by using advanced encoding and decoding methods to increase speed.

2. Overview

The throughput rate of BCH encoding and decoding must increase as the throughput rate for a device increases and BCH throughput rate must be maintained as the number of errors per sector increases with density and sector length increases. For a software BCH code implementation, performance is influenced by encode speed and decode speed with and without errors. If errors are relatively rare, performance is dominated by encode speed and error free decode speed. The error correction time for more than a few errors is typically dominated by syndrome computation and root finding. Syndrome computation time can be reduced by first computing a remainder and then computing syndromes from the remainder. Root finding time can be reduced by using root finders that are faster than the well known classical Chien search.

Free BCH source code will soon be posted.

A new free “C” language software program will soon be posted on the ChannelScience web site [6] (“C” project “FastBchEnDecR400”). This program contains fast “C” functions for a flexible programmable software binary BCH encoder and decoder. Speed is achieved by implementing fast encoding, fast error free decoding, and fast error correction. Encoding speed is achieved by using an encode table to implement a parallel encoder. The parallel approach accomplishes the equivalent of shifting a shift register eight shifts at a time by fetching one vector from the encode table. To be clear, eight shifts at a time are accomplished regardless of finite field size.

3. Design Features

Fast error free decoding on read is accomplished by using the same encode table and parallel approach to compute a remainder. If the remainder is all zeros then it is assumed that no errors exist and decoding is complete. If the remainder is nonzero then syndromes are computed from the remainder and error correction is performed. For t=14 (t is the maximum number of errors correctable by the code) and GF(2^14) this BCH code software can decode 1024 byte (plus redundancy) error free sectors at the approximate average rate of 14,035 sectors per second on a 2.67 GHz i7 920 processor.

For Windows based applications that run on a standard PC and where lots of memory is available a further speedup for encoding and error free decoding is possible by processing more than eight bits in parallel by using an even larger table.

Root-finding time is the dominate contributor to error correction time.

Since root finding time is a dominate contributor to error correction time, error correction speed can be increased by implementing fast root finders. The "C" project "FastBchEnDecR400" offers an option for selecting between two root finders. The first is a fast Chien search. The second is the Berlekamp Trace Algorithm (BTA)[1,2,3,5]. The BTA Algorithm is much faster but also more complicated than the fast Chien search.

One measure of speed for a root finding method is the number of multiplies required by the algorithm. The two root finders of "FastBchEnDecR400" were compared and the results are listed in Table 1 below.

Table 1. Average Number of Finite Field Multiplications Required for the Berlekamp Trace Algorithm (BTA) Root Finder and for a Fast Chien Search

	512 Data Bytes (t=14, m=13)	1024 Data Bytes (t=14, m=14)	1024 Data Bytes (t=25, m=14)
BTA			
Finite field add count	2,488	2,131	6,362
Finite field multiply count	2,727	2,267	6,710
Fast Chien search			
Finite field add count	27,461	53,123	103,500
Finite field multiply count	27,461	53,123	103,500

Based on the number of multiplies required, the speed performance of the BTA root finder is already very impressive. However, for a software or firmware implementation a further performance improvement can be achieved by using large tables in memory to speed up the multiplies. Large tables may be acceptable for some device applications. For applications that run on a Windows based PC, the tables can be even larger for an even greater speedup. However, the speedup we get by using large tables for the multiplies of BTA is not nearly as great as the speedup we get by switching from the Chien search to the BTA algorithm in the first place.

The speedup seen by using large look-up tables is not nearly as great that seen by switching from the Chien search to the BTA.

Other fast root finding methods are known. Special very fast root finding algorithms for error locator polynomials of degree one through six are known. Other fast algorithms for finding the roots of error locator polynomials of arbitrary degree are also known. Some are limited to very special finite fields. The BTA algorithm is the fastest root finding algorithm for **arbitrary** degree polynomials that I have implemented.

4. Overview of the Berlekamp Trace Algorithm (BTA)

The BTA as defined by Berlekamp, splits the polynomial to be factored into two factors using a polynomial greatest common divisor (gcd) function. Each resulting factor is also split into two factors and so on until there exist only degree one factors. To split a polynomial the greatest common divisor function is performed on the polynomial and a trace polynomial that has as its roots about half the elements of the finite field employed.

Stop splitting polynomial factors when the degree of a factor falls below a threshold

It is faster to stop splitting when the degree of a factor falls below a threshold and instead to find the roots of such factors by even faster methods for low degree polynomials. In “FastBchEnDecR400” I stop splitting at degree four for even “m” (“m” or GF(2^m)) and at degree two for odd “m”. I am currently using special root finding algorithms for linear, quadratic, cubic and quartic polynomials. Perhaps more time could be squeezed out of root finding by stopping the splitting process at degree six or less by using special root finding algorithms for quintic and sextic polynomials as well.

5. Hybrid BCH Systems

In hybrid BCH systems, write encoding and read remainder computation are performed in hardware and syndrome computation and error correction are performed in software or firmware. In an alternative strategy, write encoding and read syndrome computation are performed in hardware and only error correction is performed in software of firmware. Hybrid BCH systems are used when software BCH systems are not fast enough. In either hybrid strategy, software error correction must be very fast and therefore the BTA root finder algorithm is a good match. A software Chien search would be too slow for many hybrid BCH systems.

6. Conclusion

The algorithms implemented within the free “C” project “FastBchEnDecR400” show that it is possible to achieve significant speed in a software binary BCH code encoder and decoder. This code set achieves speed by using a large table to accomplish parallel encoding and remainder generation and by using the BTA algorithm for fast root finding.

It may be possible to extend the life of software and hybrid BCH systems for flash memory by using functions such as those implemented in the free "C" project "FastBchEnDecR400" [6].

7. References

My references for the BTA algorithm are a paper [5] by Berlekamp and three papers [1,2,3] that discuss a BTA derivative algorithm called BTZ. In the "FastBchEnDecR400" program I implemented much of the math from BTZ, but instead of implementing the special root finding methods of low degree polynomials by Zinoviev, I implemented alternative methods. References [1-5] can be found on the internet, but there may be a fee for some of them.

Register at www.ChannelScience.com to be notified of availability of FREE C source code for BCH ENDECs.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

1. V. Herbert. Efficient root finding of polynomials over fields of characteristic 2, WEWoRC 2009, INRIA Paris-Rocquencourt.

2. V. Herbert. Efficient root finding of polynomials over fields of characteristic 2, INRIA Paris-Rocquencourt.

3. B. Biswas, V. Herbert. Efficient root finding of polynomials over fields of characteristic 2, CRI INRIA Paris-Rocquencourt.

4. V.A. Zinoviev. On the solution of equations of degree 10 over finite fields GF(2^q). In Rapport de recherche INRIA 2829, 1996.

5. E. Berlekamp (1970), Factoring polynomials over large finite fields, Mathematics of Computation, v. 24, 1970, pp. 713-735.

6. ChannelScience web site www.ChannelScience.com – FREE downloadable software: FastBchEnDecR400.

7. N. Glover and T. Dudley, *Practical Error Correction Design for Engineers* - Revised Second Edition, Cirrus Logic, 1991.

About the Author

Neal Glover was a driver in moving the hard disk drive industry to more powerful error correction codes in the early 1990s. He has been interested in the practical application of error correcting codes for more than 30 years.

Neal has taught short courses on the subject and holds a number of patents in the field. He started two small businesses to provide error correction products and services to the magnetic and optical storage industries.

He enjoys programming in MATLAB® and "C" and has developed an extensive finite field function library for prototyping error correction algorithms in MATLAB®.