Binary BCH Encoder/Decoder

MT29Fxxxxxxx

Program Description

This document describes the binary BCH encoder/decoder for NAND Flash memory. NAND Flash memory sectors consist of a 512-byte data area and a 16-byte spare area for 528 total bytes per sector. A binary t-error-correcting BCH code is implemented to ensure data integrity. The Galois field dimension is m = 13. Inputs and outputs to or from the encoder/decoder are all in byte format. Data is read out in 2-hex words per byte and the information length of the BCH code must be a multiple of 4.

To accelerate the encoding/decoding process, parallel processing is implemented in the encoding and syndrome-computation procedures.

Program Modules

All program modules are developed with C language, taking advantage of its portability and extensive compatibility. The program modules are detailed below.

Galois Field and Generator Polynomial

The Galois field and generator polynomial are defined for global use.

Table 1: Galois Field Generator Input

Function	Code String
Program	`bch_global.c.`

Random Data Generator

Random data can be generated as input to the encoder. Default output is the typical 11112222… pattern in hex format. Random mode generates random hex data and is enabled by switch -r.

Table 2: Random Data Generator Inputs

Function	Code String
Program	`data_generator.c.`
Syntax	`data -n [byte length] -r > output file.`
Switches	`-n [byte length]: length of data in bytes;`
	`-r: random mode.`

BCH Encoder

The information data is k = 512 bytes (4,096 bits). The encoder reads the data in hex format and encodes systematically. Parallel encoding is employed for high-speed processing. The codeword output is in hex format.

Codeword structure: information data + parity check bytes. Zeros are padded to form byte-wise codewords.

Table 3: BCH Encoder Inputs

Function	Code String
Program	bch_encoder.c.
Syntax	bch_encoder -m [field] -k [info length] -t [correcting] -v < input file > output file.
Switches	-m [field]: dimension of Galois field;
	-k [info length]: information bit length, must divide 4;
	-t [correcting]: correction capability of BCH code;
	-v: verbose mode, output detailed information.

Random Error Generator

Random errors can be applied to the codeword by flipping corresponding bits.

Table 4: Random Error Generator Inputs

Function	Code String
Program	error.c.
Syntax	error -e [error number] < input file > output file.
Switches	-e [error number]: the number of errors that corrupt the data.

BCH Decoder

The decoder reads in data blocks in hex format and computes the syndrome polynomial and syndrome values in parallel. To find the error-locator polynomial, apply the simplified Berlekamp Massey algorithm. Finally, run Chien's search for error locations and bit flipping.

Table 5: BCH Decoder Inputs

Function	Code String
Program	bch_decoder.c.
Syntax	bch_decoder -m [field] -k [info length] -t [correcting] -s -v < input file > output file.
Switches	-m [field]: dimension of Galois field;
	-k [info length]: information bit length, must divide 4;
	-t [correcting]: correction capability of BCH code;
	-s: output corrected parity bits;
	-v: verbose mode, output detailed information.

Test Script

The "test_script" runs all modules and stores output information to files in the same directory.

Data Generation Script:

```
data -n 64 > data_in.txt
Output: 1111222233334444;
```

BCH Encoder Script:

```
bch_encoder -m 8 -k 64 -t 4 < data_in.txt > data_codeword.txt
Output: 11112222333334444906390C26;
```

Error Script:

```
error -e 24 < data_codeword.txt > data_error.txt
Output: 91112222337B444490639C26;
```

BCH Decoder Script:

```
bch_decoder -m 8 -k 64 -t 4 < data_error.txt > data_out.txt
Output: { Codeword 1: 3 errors found at location: 0 41 44}
        1111222233334444.
```

Algorithm Descriptions

Generator Polynomial

The generator polynomial g(x) contains 2t consecutive zeros $\alpha, \alpha^2, \ldots\ldots, \alpha^{2t}$, where α is the primitive element in GF(2^m). g(x) can be computed as:

$$g(x) = LCM\{M_1(X), \ldots\ldots, M_{2t}(x)\}$$

$$= LCM\{M_1(x), M_3(x), \ldots\ldots, M_{2t-1}(x)\}$$

where $M_i(x)$ is the minimal polynomial defined by the cyclotomic coset and LCM represents the least common multiple.

Reference [1]: Lin and Costello, Chapter 6.1.

Encoder

Let d(x) be the information data; s(x) be the parity check bits; and c(x) be the codeword. The systematic encoder takes the following encoding process,

$$s(x) = x^{n-k}\, d(x)\ mod\ g(x)$$

$$c(x) = s(x) + x^{n-k}\, d(x)$$

Reference [1]: Lin and Costello, Chapter 5.3.

Syndrome Computation

Let r(x) be the received data and s(x) be the syndrome polynomial. The syndrome polynomial is computed via:

$$s(x) = r(x) \ mod \ g(x)$$

and the 2t syndromes can be computed via

$$S_i = s(\alpha^i)$$

Reference [1]: Lin and Costello, chapter 6.2.

Berlekamp Massey Algorithm

The BMA starts with 2t syndrome values S_1, S_2, ..., S_{2t}. The simplified BMA for binary BCH code skips the even steps since the discrepancy in even steps is always zero. Thus, only t iterations are needed instead of 2t iterations.

Algorithm Description

1. Initialization:

$$\Lambda^{(1)}(x) = 1 + S_1 \, x, \, L_1 = 1, \, B^{(1)}(x) = S_1^{-1}$$

2. Do t-1 iterations for odd syndromes, r = 3, ..., 2t-1 as follows:

- Compute discrepancy Δ_r;

$$\Delta r = \sum_{i=1}^{L_{r-2}} \Lambda_i^{(r-2)} \ S_{r-i} + S_r$$

- If $\Delta_r = 0$,

$$\Lambda^{(r)}(x) = \Lambda^{(r-2)}(x)$$

and

$$L_r = L_{r-2}$$

then

$$B^{(r)}(x) = x^2 \ B^{(r-2)}(x)$$

- If $\Delta_r \neq 0$,

$$\Lambda^{(r)}(x) = \Lambda^{(r-2)}(x) - \Delta_r \, x^2 \ B^{(r-2)}(x)$$

- If $2 L_{r-2} \geq r$, and

$$L_r = L_{r-2}$$

then

$$B^{(r)}(x) = x^2 B^{(r-2)}(x)$$

- If $2 L_{r-2} < r$, and

$$L_r = r - L_{r-2}$$

then

$$B^{(r)}(x) = B^{(r-2)}(x)/\Delta_r$$

Reference [2]: Blahut, Chapter 7.6.

Chien's Search

Chien's search substitutes all the elements in the Galois field into the error locator polynomial $\Lambda^{2t-1}(x)$. If α^i is a root of $\Lambda^{2t-1}(x)$, then α^{n-i} is the error location. Because binary BCH codes only have bit-flipping errors, after finding the error location, it is easy to correct the error by flipping the error bit.

Reference [1]: Lin and Costello, Chapter 6.5.

Parallel Encoding/Syndrome Computation

Based on the generator polynomial, a lookahead table can be constructed to process the data in parallel.

Reference [3]: Shieh.

References

[1] S. Lin and D. J. Costello, *Error Control Coding,* 2nd ed. New York: Prentice Hall, 2004.

[2] R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.

[3] M. Shieh, M. Sheu, C. Chen, and H. Lo, "A systematic approach for parallel CRC computations," *Journal of Information Science and Engineering*, May 2001.

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-3900
prodmktg@micron.com www.micron.com Customer Comment Line: 800-932-4992

Revision History

Rev. A 8/06

- Initial release.