ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Design of Flash Controller for Single Level Cell NAND Flash Memory

Ashwin Bijoor [1], Sudharshana [2]

P.G Student, Department of Electronics and Communication, NMAMIT, Nitte, Karnataka, India [1]

Assistant Professor, Department of Electronics and Communication, NMAMIT, Nitte, Karnataka, India [2]

ABSTRACT: NAND flash is used in many modern memory applications. The paper presents a novel approach for design of NAND flash controller. An 8 bit error correcting Reed Solomon (RS) is designed for detecting and correcting errors of flash memory. RS encoder is designed using cyclic encoder design. RS decoder used in controller design is complex. Decoder design uses Berlekamp Massey, Chien Search and Foney algorithms for detecting and correcting errors. Both RS encoder and decoder designs are efficient and utilizes Field Programmable Gate Array (FPGA) hardware effectively. Controller design also proposes an effective method for generating control signals for NAND flash memory.

KEYWORDS: Berlekamp Massey, Chien Search and Foney algorithms

I. INTRODUCTION

Flash memory is a non-volatile memory that is used in applications where high memory designs are required. NAND flash memory must be accessed sequentially. Therefore a controller is designed for sequential access of data. The block diagram of NAND flash controller is shown Figure 1.

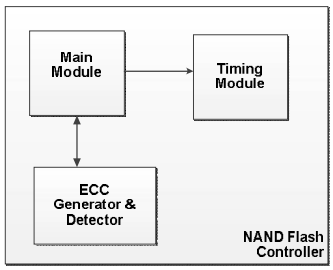

Figure 1: NAND Flash Controller

NAND flash controller block controls flow of data from memory to host or from host to memory using main module. Module also generates control signals and set of commands required for exchange of data. Timing module monitors time delay to data and control signals. It ensures that data and control signals are applied to NAND flash when it is in ready state. Timing module produces required amount of delay to corresponding control signals so that required operation is successfully performed. Error Correction code (ECC) generator and detector module generates Cyclic Redundancy Codes (CRC) for data to be written into flash, detects and corrects error in data block that is read from flash memory.

II. LITERATURE SURVEY

Most of NAND flash application use hamming code for detecting correcting errors. But hamming code can only correct single error in a block of data. In case of applications where each bit of data is very critical, this could lead to failure of the device. So RS encoder design is used to provide a reliable and efficient way of producing correction bits. The

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

correction bits produced at output provides capability of correcting multiple errors. The encoder design uses cyclic encoder design to produce correction bits [1], [7]. These correction bits are produced using two arithmetic operations that is galios multiplication and galios addition [2]. The RS decoder design proposed detects and corrects error. The decoder design uses various blocks. The block used in decoder design are syndrome computation, Berlekamp Massey algorithm, Chien Search algorithm [5], foney algorithm [6] and error correction. The controller designed uses an asynchronous interface for NAND flash to speed up communication.

III.RS ENCODER

RS encoder is represented as RS(m, k) with m bit symbols and t symbol error correction capability [1]. RS encoder output is m symbol block for k symbol data. Figure 3 shows RS encoder block diagram. A function of the transmitted message m(x), the generator polynomial g(x) and the number of parity symbols 2t is defined and systematically encoded as transmitted code word as shown in (1) [7], [1].

$$c(x) = m(x) \cdot 2t + m(x) \cdot mod\ g(x) \qquad (1)$$

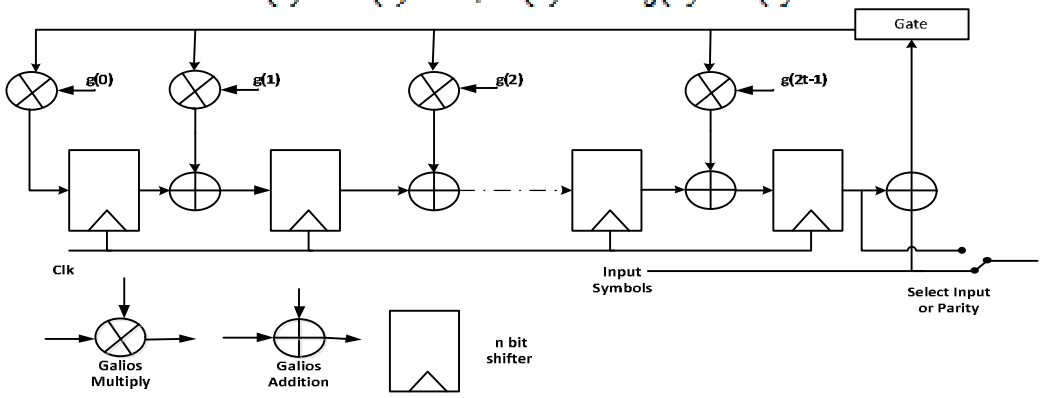

Figure 3: RS Encoder Block Diagram

A. Galois Multiplier

The Galois multiplication of two 8 bit data is done using (2) to (19).

$$\beta = \sum_{i=0}^{7} a_i \cdot \alpha^i \qquad (2)$$

$$\gamma = \sum_{i=0}^{7} b_i \cdot \alpha^i \qquad (3)$$

$$\beta \cdot \gamma = \sum_{i=0}^{14} p_i \cdot \alpha^i \qquad (4)$$

Intermediate product p can be obtained multiplying β and γ and equating equal power terms.

$p_0 = a_0 \cdot b_0$ (5)
$p_1 = a_0 \cdot b_1 + a_1 \cdot b_0$ (6)
$p_2 = a_0 \cdot b_2 + a_1 \cdot b_1 + a_2 \cdot b_0$ (7)
$p_3 = a_0 \cdot b_3 + a_1 \cdot b_2 + a_2 \cdot b_1 + a_3 \cdot b_0$ (8)
$p_4 = a_0 \cdot b_4 + a_1 \cdot b_3 + a_2 \cdot b_2 + a_3 \cdot b_1 + a_4 \cdot b_0$ (9)
$p_5 = a_0 \cdot b_5 + a_1 \cdot b_4 + a_2 \cdot b_3 + a_3 \cdot b_2 + a_4 \cdot b_1 + a_5 \cdot b_0$ (10)
$p_6 = a_0 \cdot b_6 + a_1 \cdot b_5 + a_2 \cdot b_4 + a_3 \cdot b_3 + a_4 \cdot b_2 + a_5 \cdot b_1 + a_6 \cdot b_0$ (11)
$p_7 = a_0 \cdot b_7 + a_1 \cdot b_6 + a_2 \cdot b_5 + a_3 \cdot b_4 + a_4 \cdot b_3 + a_5 \cdot b_2 + a_6 \cdot b_1 + a_7 \cdot b_0$ (12)
$p_8 = a_1 \cdot b_7 + a_2 \cdot b_6 + a_3 \cdot b_5 + a_4 \cdot b_4 + a_5 \cdot b_3 + a_6 \cdot b_2 + a_7 \cdot b_1$ (13)
$p_9 = a_2 \cdot b_7 + a_3 \cdot b_6 + a_4 \cdot b_5 + a_5 \cdot b_4 + a_6 \cdot b_3 + a_7 \cdot b_2$ (14)

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

$p_{10} = a_3 \bullet b_7 + a_4 \bullet b_6 + a_5 \bullet b_5 + a_6 \bullet b_4 + a_7 \bullet b_3$ (15)
$p_{11} = a_4 \bullet b_7 + a_5 \bullet b_6 + a_6 \bullet b_5 + a_7 \bullet b_4$ (16)
$p_{12} = a_5 \bullet b_7 + a_6 \bullet b_6 + a_7 \bullet b_5$ (17)
$p_{13} = a_6 \bullet b_7 + a_7 \bullet b_6$ (18)
$p_{14} = a_7 \bullet b_7$ (19)

Output bit p_0 to p_{14} is intermediate product bits. These bits are used to calculate Galois product.

$d_0 = p_0 + p_8 + p_{12} + p_{13}$ (20)
$d_1 = p_1 + p_8 + p_9 + p_{12} + p_{14}$ (21)
$d_2 = p_2 + p_9 + p_{10} + p_{13}$ (22)
$d_3 = p_3 + p_8 + p_{10} + p_{11} + p_{12} + p_{13} + p_{14}$ (23)
$d_4 = p_4 + p_8 + p_9 + p_{11} + p_{14}$ (24)
$d_5 = p_5 + p_9 + p_{10} + p_{12}$ (25)
$d_6 = p_6 + p_{10} + p_{11} + p_{13}$ (26)
$d_7 = p_7 + p_{11} + p_{12} + p_{14}$ (27)

The outputs d_0 to d_7 represents Galois product output for 8 bit Galois input [2].

B. Generator Polynomial

The generator polynomial is generated using primitive polynomial and primitive element α. p(x) = x8+x4+x3+x2+1 is used as primitive polynomial. The value of primitive element used is 2 (i.e. α = 2).

The generator polynomial is derived by applying Galois multiplication and Galois addition for a set of variables. Generation of generator polynomial for RS coding with 8 bit errors is shown in (28) and (29) [1].

$$g(x) = \prod_{i=0}^{15} (x - \alpha^i) \qquad (28)$$

$$g(x) = x^{16} + 110\,x^{15} + 252\,x^{14} + x^{13} + 196\,x^{12} + 127\,x^{11} + 98\,x^{10} + 241\,x^9 + 231\,x^8 + 48\,x^7 + 61\,x^6 + 126\,x^5 + 60\,x^4 + 210\,x^3 + 193\,x^2 + 142x + 29 \qquad (29)$$

IV. RS DECODER

Decoding process is difficult to understand and to implement in hardware than encoding process [3]. Figure 4 shows RS decoder implementation diagram.

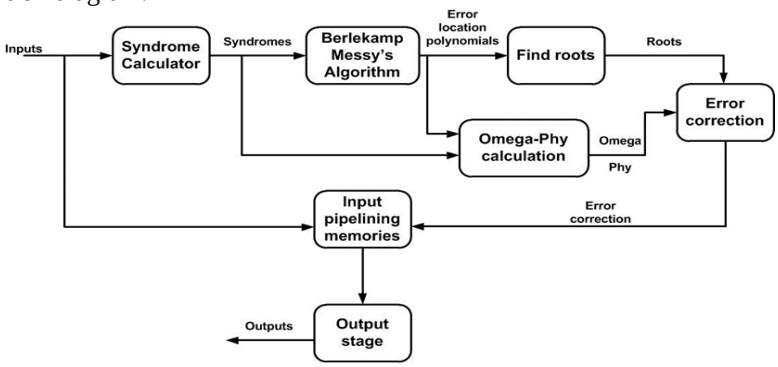

Figure 4: RS Decoder Implementation Diagram [4]

A. Syndrome Computation

At the receiver, syndrome is calculated to check presence of errors. The syndrome polynomial contains point and proportion of upto t errors in an invalid code word [4]. The computation of syndrome coefficients is given by (30).

$$S_i = R(x)|_{x=\alpha^i} = R(\alpha^i) \qquad (30)$$

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Thus syndrome polynomial is represented as

$$S = \sum_{i=0}^{i=x^{2t-1}} S_i \bullet x^{(i-1)} \qquad (31)$$

where R(x) is received symbols and it is represented as

$$R(x) = \sum_{n=0}^{n=2t-1} R_{n-1} \bullet x^{(n-1)} \qquad (32)$$

B. Berlekamp Massey Algorithm

Key equation solver is the main component of RS decoder. This equation involves 2t linearly dependent equations. It generates the key equations [5].

$\Phi(x) \bullet S(x) = \Omega(x) \bullet \text{mod}(x2t)$ (34)

where $\Phi(x)$ is error locator polynomial and $\Omega(x)$ is error evaluator polynomial [6].

C. Chien Search Algorithm

Chien search performs function of finding error locations, when the Chien sum is zero [6]. Chien search utilizes all possible input variables and then checks whether outputs are zero. The summation of odd variables are computed at one end and the summation of even variables is computed at other end [5]. Then two summation outputs are added. If summation variable is zero at any clock cycle, then error point is determined from position of clock cycle [6]. The magnitude of error value is calculated using Forney Algorithm. The coefficients of $\Omega(x)$ is input for algorithm [6].

D. Error correction

Once error points and proportions are computed, error correction block accepts received code word. At particular error location, XOR operation is performed with received code word and error magnitude at the location to obtain corrected massage symbols. To interlock order of bytes in both variables, a FIFO register is utilized either at received code word or at error variable because error variables are produced in reverse order of received code word [6].

V. NAND FLASH CONTROLLER

Figure 3.1 shows block level representation of NAND Flash Controller. The control signal output is generated using a 50MHz clock at input of Controller. On application of reset pulse at the input, all variables are initialized to default value. After a reset pulse, an activate signal is made logic high. At the occurrence of activate signal, controller gets activated and waits for a command input that specifies controller about operation to be performed on flash. Depending on operation, control signals are generated and are applied to flash. The controller also has two 8 bit data I/O that interfaces both flash as well as PC.

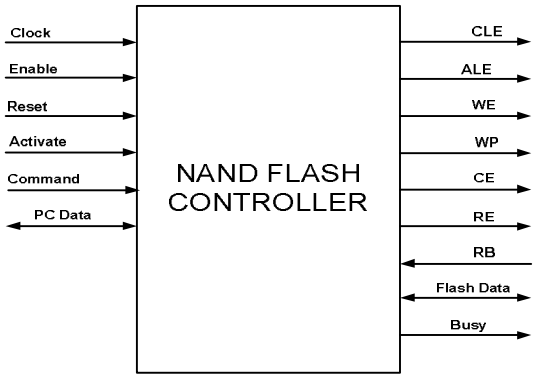

Figure 5: NAND Flash Controller

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

VI. RESULTS AND DISCUSSION

Figure 6 shows an example control signal generation for reset operation. Initially nreset goes logic low state to initialize all variables. Once nreset goes logic high, an activate signal is given for a clock pulse. On applying activate pulse, command FFh is latched onto data bus for one clock cycle.

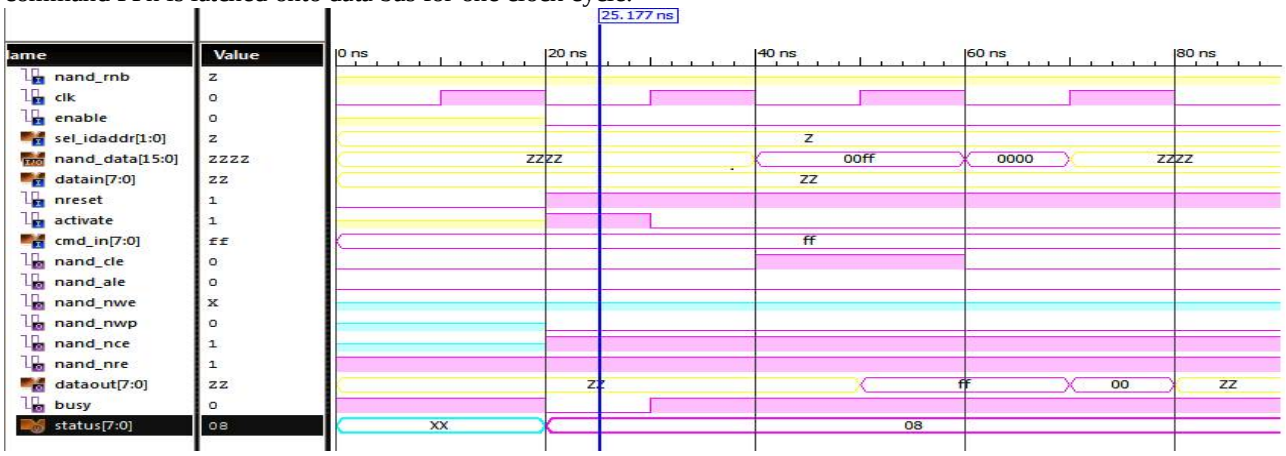

Figure 6: Flash Control Signal Generation

Figure 7 shows RS encoder simulation results. During first two clock cycles, all variables are initialized and a counter k is initialized to 0. After third pulse, input is applied to encoder through datain port and dataout port is switched to input. Once counter reaches a value of 188, the dataout is switched to output of encoder.

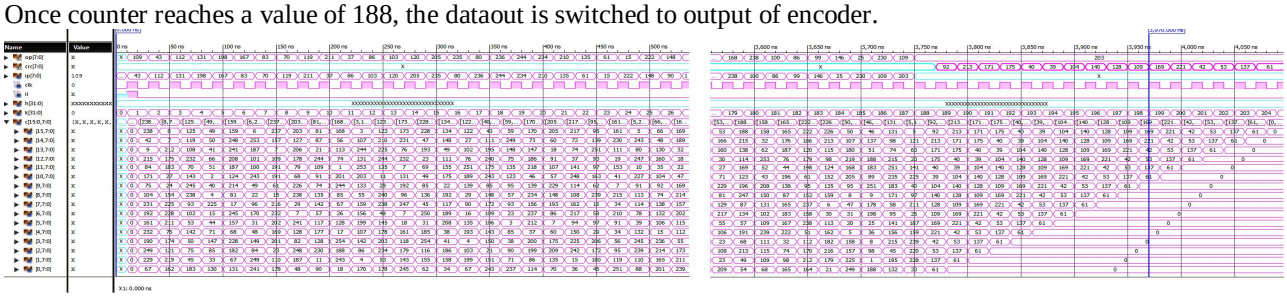

Figure 7: RS Encoder Results

Figure 8 shows RS decoder simulation. During first few clock cycles, inputs are accepted, syndrome values are calculated and stored into a buffer. The syndrome values calculated are applied to other blocks in consecutive clock cycles to deduce error position and magnitude of error. In simulation, Address_error shows that there is an error at location number 131 and correction_value shows magnitude of error as 180.

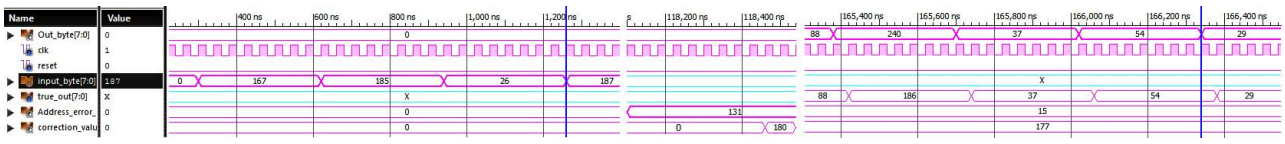

Figure 8: RS Decoder Results

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Table 1 shows synthesis report of different blocks in NAND flash controller.

Table 1: Synthesis Report of NAND flash controller

Functional Blocks	Number of Slices	Number of Flip Flops	Number of LUT's	Number of IOB's	Minimum Period (ns)	Power (mW)	Maximum Frequency (MHz)	Memory Usage
RS Encoder	27	0	52	24	10.244	60	97.618	240960 kB
RS Decoder	3076	2826	5675	21	12.557	60	79.637	406336 kB
Control Signal Generation	2446	2526	2788	52	11.179	60	89.456	336192 kB

VII. CONCLUSION

A novel design of NAND flash controller was presented to control operations on flash memory. An 8 bit error correcting RS encoder and decoder was designed and simulated using Verilog on Xilinx ise 13.2. The analysis of result shows that encoder produces CRC bits within 4.2 µs, decoder detects and corrects error within 150 µs. Another important part of flash controller is control signal generation. The synthesis report shows that FPGA hardware is efficiently utilized and flash controller can be effectively implemented.

REFERENCES

[1] Azad, Aqib Al, and Md Imam Shahed. "A Compact And Fast FPGA Based Implementation Of Encoding And Decoding Algorithm Using Reed Solomon Codes". IJFCC, pp. 31-35, 2014.
[2] Lin, Shu, and Daniel J Costello. Error Control Coding. Englewood Cliffs, N.J.: Prentice-Hall, 1983.
[3] Tiwari, Bhawna, and Rajesh Mehra. "Design And Implementation Of Reed Solomon Decoder For 802.16 Network Using FPGA", IEEE International Conference on Signal Processing, Computing and Control, 2012.
[4] Dayal, Priyanka, and Rajeev Kumar Patial. "FPGA Implementation Of Reed-Solomon Encoder And Decoder For Wireless Network 802. 16". International Journal of Computer Applications, pp. 42-45, 2013.
[5] Sarwate, D.V., and N.R. Shanbhag. "High-Speed Architectures For Reed-Solomon Decoders". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 641-655, 2001.
[6] Das, Anindya Sundar, Satyajit Das, and Jaydeb Bhaumik. "Design Of RS (255, 251) Encoder And Decoder In FPGA". International Journal of Soft Computing and Engineering, vol. 2, pp. 391-394, 2013.
[7] Ziliang Lin, Changyin Liu, and Lei Liu. "An Efficient RS Encoder For CCSDS". 2013 IEEE 4th International Conference on Software Engineering and Service Science, pp.791-794, 2013.