International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 04, April-2015

Write Pattern Format Algorithm (WPFA) for Reliable NAND-based SSDs

Girish Kumar M S
VLSI &Embedded System
SJBIT
Bangalore, India

Chetana R
Associate Professor
ECE Dept. SJBIT
Bangalore, India

***Abstract*— This paper implements and evaluates a pre-coding algorithm to improve data retention and reduce power consumption in NAND-based solid-state drives (SSDs). Write pattern format algorithm (WPFA) consumes less power while achieves better data retention when compared to the stripe pattern elimination algorithm (SPEA) and state-of-the-art asymmetric. BCH (15, 7) encoder and decoder used detects and corrects any 2 bit error in any position of 15 bit code word. BCH (15, 7) decoder is implemented using the Chine's search algorithm and Peterson algorithm. The Multichannel NAND Flash Memory Controller exploits the parallelism of multiple flash chip and executes parallel requests simultaneously by the crossbar switch operation. By which improved bandwidth utilization and the speed of accessing is achieved. The performance of WPFA, BCH (15, 7) and Multichannel NAND Flash Memory Controller is evaluated by both FPGA implementation and computer simulations.**

Keywords—Solid-state drives; NAND flash memory; BCH Encoder; BCH Decoder; Bandwidth;

I. INTRODUCTION

NAND flash memory is most popular storage technology for solid-state drivers (SSDs) due to low-power consumption, light weight, and non-volatility. Each NAND flash cell consists of a floating gate transistor whose threshold voltage can be programmed by injecting certain amounts of charge into the floating gate [1]. Gain or loss of charge occurring on the floating gate over time will lead to bit flipping and retention failures. Experiments have suggested that bit flipping errors are asymmetric and not random; specially "1 → 0" errors are dominant in the upper pages. and "0 → 1" errors occur in the lower pages .Hence to reduce the retention error rate, it is useful to distribute more 0's to upper pages while more 1's to lower pages[2].

Besides retention reliability, another practical problem is the increasing power required for scaling to larger bit-line capacitances. In SSDs, more power is being consumed to charge or discharge the parasitic capacitance of the bit-lines (BLs). Several approaches have been proposed to address the problems of data retention and power [2]. Among these, asymmetric coding and the stripe pattern elimination algorithm (SPEA) perform well by processing data patterns. In the first step, SPEA calculates the difference between the numbers of 1's in even and odd columns of the original data. If the difference is higher than a threshold value, bits will be rearranged to eliminate the CSPs, which relieves the power problem. Secondly, asymmetric coding calculates the number of 1's in the input data which is then used to determine whether the bits within the unit are flipped or not. As a result, the distribution of 1's becomes asymmetric [3].

Even though asymmetric coding and SPEA improve SSD performance considerably, their implementation is complex especially when the code length of SPEA increases [4]. Meanwhile, additional CSPs are introduced due to the arrangement during the SPEA processing which could cause power problems as well. In this paper, our goal is an efficient solution to the problems above with low complexity implementation. We first present a write pattern format algorithm (WPFA) that carries out asymmetric processing and stripe elimination simultaneously, which allows data patterns to be modified only once before being fed to an error correction coding (ECC) module. Here it should be noted that the advantages of the solution comes at the cost of a small loss of performance compared to asymmetric coding. WPFA will achieve an improvement over the original SPEA approach by avoiding the extra CSPs introduced in SPEA and reducing power consumption [5].

The rest of this paper is organized as follows. The basic principle of working of WPFA, BCH Encoder and Decoder are discussed in section 2. Block by block explanation of Framework of coding mechanism in SSDs is discussed section 3. Final conclusion remarks are given in section.

II. WORKING OF WPFA & BCH

A. Write Pattern Format Algorithm

WPFA has two primary stages. The first stage modifies the program data to eliminate the column stripe patterns; the second stage increases the number of 1's. Note that the length of data processing unit has been restricted to 2n with n = 2, 3, 4, 5 . . . Initially, all bits of the data unit are added together and the sum is stored in an n-bit sum register. The most significant bit (MSB) of the sum register is used as the flag, indicating whether the majority of bits in the data unit are 0 or 1. WPFA eliminates CSPs in the following way. If the flag equals 0, the data unit is passed unmodified to the next processing stage. If the flag equals 1, a column stripe sequence is added to the input data. The modulo-2 result is taken as "first stage data", in other words, half of the input data will be flipped. As a result of this stage, column stripe patterns are eliminated.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

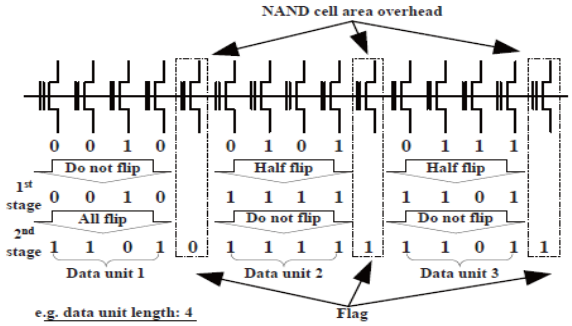

Fig. 1. WPFA.

In WPFA, retention reliability has been further improved by increasing 1's in lower and 0's in upper pages. At the second stage, the flag (MSB of the sum register) continues to be used for determining whether the "first stage data" is flipped or not. If the flag equals 0, indicating that the majority in the input data pattern are zeros, all bits of the "first stage data" are flipped. If the flag equals 1, the "first stage data" will not be modified. Consequently, the number of 1's in lower pages data increases. This part of WPFA is similar to asymmetric coding. "Second stage data" together with flag bits are used to form the output. For the data patterns processing of upper pages, the idea is the same except the goal is to decrease the number of 1's.

Fig. 1 illustrates an example of the WPFA with lower page input data. In this example, the length of data unit is 4 bits and the width of the sum register is 2 bits. At the first stage, "data unit 1" keeps unmodified while "data unit 2" and "data unit 3" are half-flipped since the flags of them are equal to 1. At second stage, only "data unit 1" is modified.

B. BCH (15, 7) Encoder & Decoder

BCH codes are defined by two parameters that are the number of errors to be corrected t and code size n.

Block length: n = 2m - 1

Number of information bits: k ≥ n-m*t

Minimum distance: dmin≥2t + 1.

The generator polynomial of the code is specified in terms of its roots over the Galois field GF (2m). Let α be a primitive element in GF (2m).The generator polynomial g(x) of the code is the lowest degree polynomial over GF (2). Let $m_i(x)$ be the minimum polynomials of α_i then generator polynomial G(x) can be computed.

$$G(x) = LCM\ [m1(x), m3(x)\ldots m2t(x)] \quad (1)$$

Here we consider n=15, k=7 and t=2. So the generator Polynomial with $\alpha, \alpha^2, \ldots \alpha^4$ by multiplying the following minimal polynomials the roots are obtained:

$$m1(x) = 1+x+x4$$

$$m3(x) = 1+x+x2+x3+x4$$

Substituting m1(x) and m3(x) in equation (1) generator polynomial is obtained.

$$G(x) = LCM\ \{m1(x), m3(x)\}$$

$$G(x) = \{(1+x+x4)\ (1+x+x2+x3+x4)\}$$

To build BCH codes over GF (24), we need to find out the elements of GF (24) generated by p(x) =1+x+x4 is given intable below.

Table -1: The Elements of GF (24) Generated By P(X) =1+X+X4.

Powers of primitive elements	Binary representation	Polynomial form
α^0	0001	1
α^1	0010	α
α^2	0100	α^2
α^3	1000	α^3
α^4	0011	$\alpha+1$
α^5	0110	$\alpha^2+\alpha$
α^6	1100	$\alpha^2+\alpha^3$
α^7	1011	$1+\alpha+\alpha^3$
α^8	0101	α^2-1
α^9	1010	$\alpha+\alpha^3$
α^{10}	0111	$1+\alpha+\alpha^2$
α^{11}	1110	$\alpha^3+\alpha+\alpha^2$
α^{12}	1111	$\alpha+1+\alpha^2+\alpha^3$
α^{13}	1101	$1+\alpha^3+\alpha^2$
α^{14}	1001	$1+\alpha^3$
α^{15}	1	1

III. HARDWARE DESIGN & IMPLEMENTATION COMPLEXITY

In this section, we consider the logic circuits for the WPFA algorithm and implement it, to reduce the hardware complexity. Generally, WPFA will be implemented together with the BCH module as part of the flash controller in FPGA. The overall framework of FPGA-based flash controller is shown in Fig. 2. The brief explanation of each block is given below.

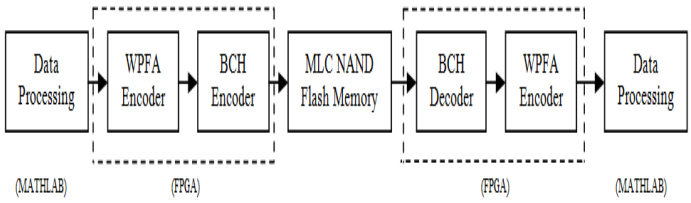

Fig. 2. Framework of coding mechanism in SSDs.

A. Data Processing

The input may be any16 bit data. The data given here is an image, but an image cannot be given directly to memory as input. Generally, the image is converted to gray scale as shown in Fig.4 and its pixel values are extracted via a MATLAB code. This pixel values can be represented in binary, hexadecimal or decimal. Once done they are given to the program as input using a block of memory depending upon size or resolution of the image. Consider an example. An image of 100*100 resolutions will use 10,000 address locations on the memory.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

Fig. 3. Original Image.

Fig. 4. Grey Scale Image & final output image.

The last block is also shown as data processing, the output of the WPFA decoder is taken and this is processed using the MATLAB code to obtain the grey scale image which was applied as input.

B. WPFA Encoder & Decoder

Fig. 5 illustrates the circuit structure for the WPFA of 16-bit data unit. Before write patterns processing, the parallel 16-bit data and upper/lower-page select signal (U/L) are generated from information bit stream by the serial/parallel converter. The number of 1's in the data unit is then calculated with 16-bit adder circuit and the MSB of the 4-bit sum register is used as the judge signal for the subsequent computations. In our implementation circuit, comparators and multiplexers have been replaced with simple XOR gates, to perform bit-flipping operations. The flag bit is created through a series of logic operations over MSB and U/L. Upper page and lower page asymmetric encoders are implemented with a separate circuit unit. These two encoders for WPFA are integrated in a single circuit which saves FPGA resources. Note that the U/L signal will be correspondingly produced when the flash controller fetches data from the memory array. Hence, in the decoding side, the decoder circuit is easy to implement by performing XOR operations over the input data, the flags and the U/L signal.

C. BCH (15, 7) Encoder

BCH Encoder is implemented with a Linear Feedback Shift Register (LFSR). (15, 7) BCH code words are encoded as follows.

$$C(y) = Y^{N-K} * N(y) + A(y) \qquad (2)$$

Where, Message bits $N(Y) = N_0 + N_1 y + \ldots + N_{K-1} y^{K-1}$.
Codeword $C(y)$ is $C_0 + C_1 y + \ldots + C_{N-1} y^{N-1}$.
Remainder $A(y) = A_0 + A_1 y + \cdots + A_{M-1} Y^{M-1}$. Also, Ci, Ji, Ai are the subsets of Galois field.

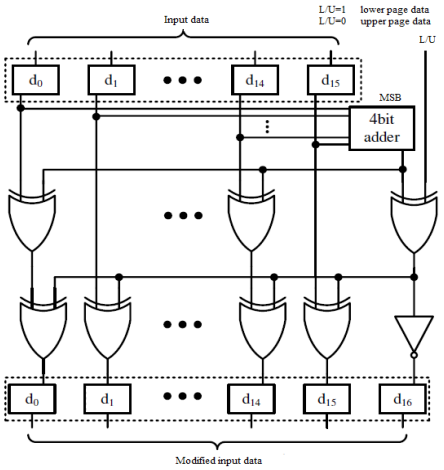

Fig. 5. Circuit schematic of the joint lower/upper pages coding.

Fig 6 shows block diagram of (15, 7) BCH Encoder module. This 7 message bits (M0, M1….M6) are fed into the parallel to serial shift register. This Serial output will be sent to (15, 7) BCH Encoder module as shown in Fig 7. The parity bits are computed using these message bits and sent to serial to parallel shift register, to append the parity bits to the original message bits to obtain 15 bit encoded data. The entire encoding process requires 15 clock cycles.

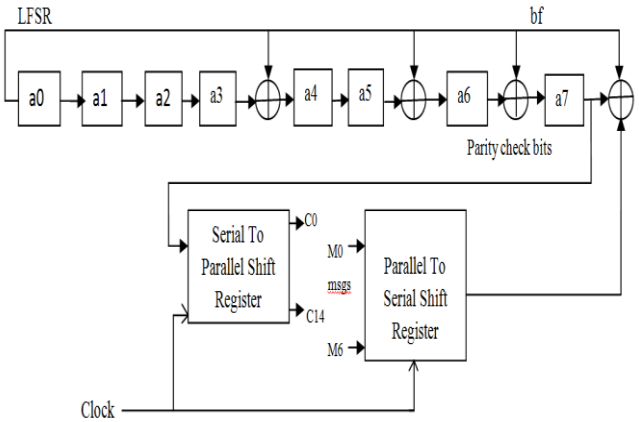

Fig. 6. Block diagram of (15, 7) BCH Encoder

D. MLC NAND Flash Controller

Controller looks after the every instruction given to flash memory as the name itself indicates, Multichannel basically exploits parallelism in a flash memory. By using multiple controllers we achieve this parallelism. The basic multichannel flash design will be as shown in the figure 7.

The host requests are directly fed into flash translation layer. This converts physical address to logical address. Multiplexers are used for the purpose of choosing the controller. Flash memory is accessed through flash controller which is chosen by multiplexer. The logical address is converted into main logic and control logic. The data and control logic contains control signals for specific function to perform in controller such as read, write and erase are present in the main logic.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

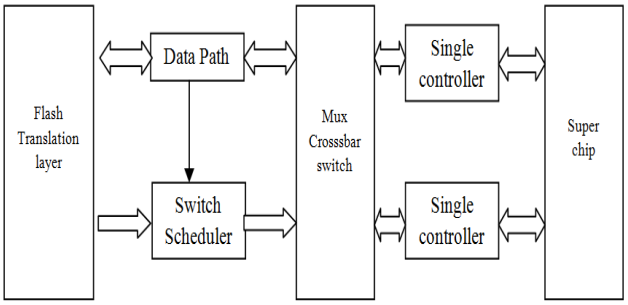

Fig. 7. Multichannel NAND Flash design

With the help of Finite State Machine (FSM) the control signal working can be seen as shown in figure 8.

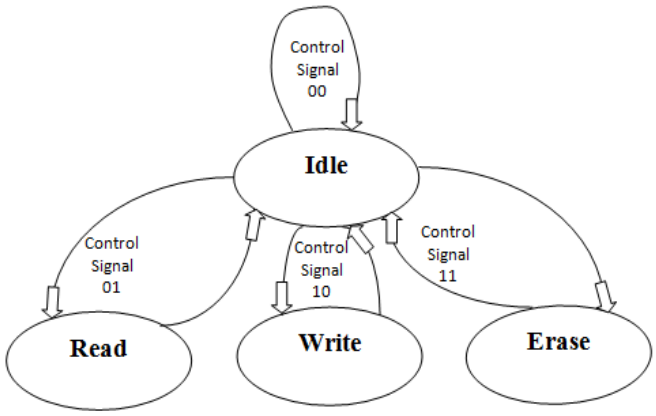

Fig. 8. Finite state machine for controller.

The controller basically will be in idle state and they will move to read, write and erase state as shown in table 2. The controller will reach the ideal state after completion of each function.

Table. 2. Controller operations

Control signal	Operation	Action
00	Idle	Controller will be in idle state
01	Read	The data will be read by host from flash memory
10	Write	The data will be written into flash memory by host
11	Erase	The block of data will be erased from flash memory

1) FTL: Flash Translation layer or FTL basically emulates as a hard disk drive which eventually translates host requests into flash requests. FTL provides some core functionalities such as address translation, bad block management, wear-levelling and error correction code checking. An FTL receives, reads and writes requests from the file system and maps a logical address to a physical address in the NAND flash. Role of an FTL is to redirect each write request to an empty area of flash memory, thereby avoiding the "erase-before-write" limitation of flash memory.

2) Crossbar Switch: As we are interested in multichannel flash memory we have to use some sort of selector, which are nothing but multiplexers. Mux will be used for selection purpose the select lines for mux will be given by switch scheduler in the block. The multiplexer will be seen as shown in the fig 9.

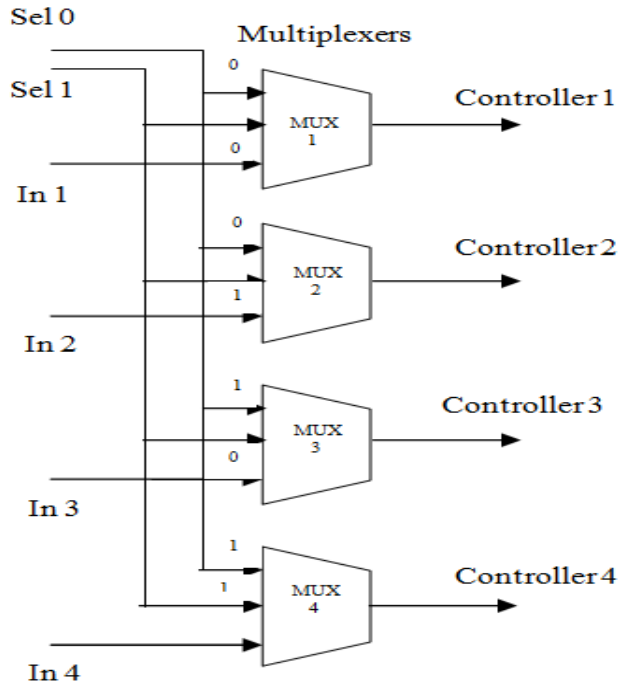

Fig. 9. Multichannel NAND Flash design

3) Flash Controller: Controller gets the data from switch scheduler and it will perform operations as per table given in table 2. The typical flash device will be as shown in the fig 10.

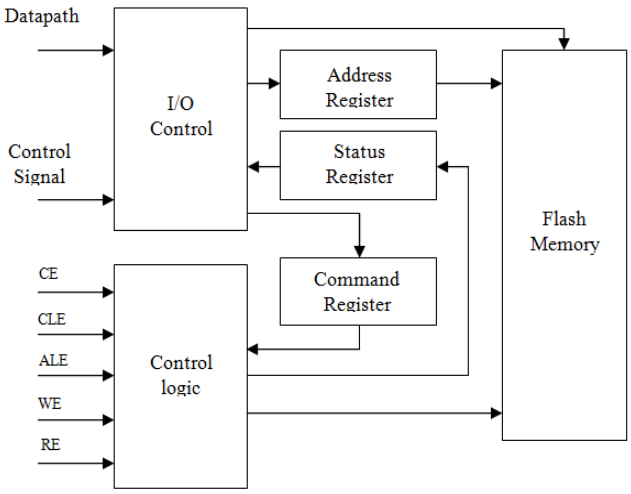

Fig.10 . Typical flash controller.

The inputs to controllers are converted inputs from Flash Translation Layer (FTL). The FTL translates host requests into flash requests, where the host requests such as read and write requests are specified in terms of logical sectors. The FTL issues the commands, addresses and data to the multichannel controller via the channels of the data path. When receiving the requests the switch scheduler sends the select signal of the crossbar switch, then switching circuit sends the request to a single controller decided by the address of the request. Then according to the type of the requests the single controller sends corresponding control signals to NAND flash chip to complete the operation like read page,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

program page, erase block. The data transfer occurs between the controller and the host through the data path.

It consists of address register, data register, command register, main logic and the control logic. It has single IO path to the memory transactions, which shares the addresses, command and data.

When a request arrives the main state machine will identify the operation from the command code, and it will sends an information to the identified operation sate. Then control goes to sub-state machine and generate control signals to perform the corresponding operation. After the complete implementation sub-state machine sends done signal to the main state machine that informs the completion of action.

The control signals to the control logic are explained as below:

a) CE: This pin indicates NAND flash chip enable. When this pin is high the NAND flash memory controller are in working mode, it reflects that the data is storing in a memory in sequential order. When it is low, the NAND Flash memory controller does not work.

b) CLE: This pin indicates NAND flash clear mode. When this pin is high the data are cleared in the main memory of NAND flash memory controller. When it is low it will not work.

c) ALE: This pin indicates NAND flash advanced latch enable when this pin is high, the NAND flash memory controller work and when it is low, the NAND flash memory controller are in latch mode.

d) WE: This pin indicates NAND flash in write operation. When it is high, data are in writing process in a write mode i.e. we can write the data at particular memory address. When it is low, it will not provide a permission to write the data.

e) RE: This pin indicates NAND flash reset. When this pin is high the NAND flash memory controller are in reset mode i.e. all operations of NAND flash memory controller stopped. When it is low, it will work properly.

E. BCH (15, 7) Decoder

Fig 11 depicts (15, 7) BCH decoder module. (15, 7) decoder has 3 major steps: The first step does the Calculation of syndrome from the received word r(y), where n=1, 2,.....,2k. In which 15 bit data is applied to the parallel to serial shift register. Syndrome S(n) is calculated from the obtained output from the parallel to serial shift register. Figure 13 shows the Syndrome computation circuit. If the calculated syndrome is Zero i.e., S(n) ==0, the transmission is error free, else transmitted message has errors.

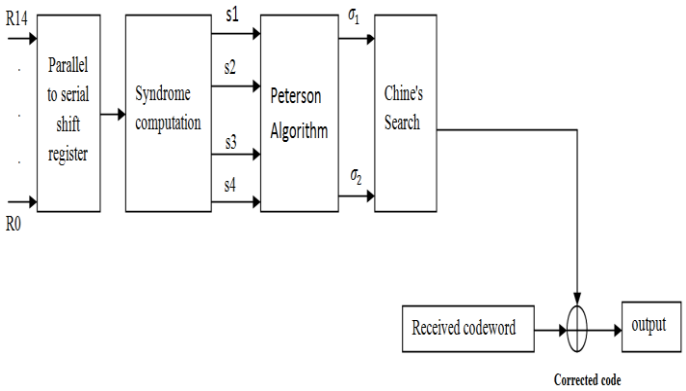

Fig.11. Block diagram for (15, 7) BCH Decoder.

The second step is Determination of error location in the syndrome. Here Peterson & Zierler's algorithm and chine's search algorithm is used in this phase Peterson algorithm is used to compute error location polynomial σ(x)in the syndrome S(n). Formula to find the error locator polynomial is $\sigma_1 = S_1, \sigma_2 = (S_3 + S_1^3) * (S_1^{-1})$.

This polynomial helps in finding out the location of the errors, Chine's search algorithm finds out the location of the error locator polynomial. The error locator polynomial is fed into chine's search algorithm to obtain roots of the error locator polynomial which corresponds to the error position. Fig 12 shows the working of double error check chine's search algorithm.

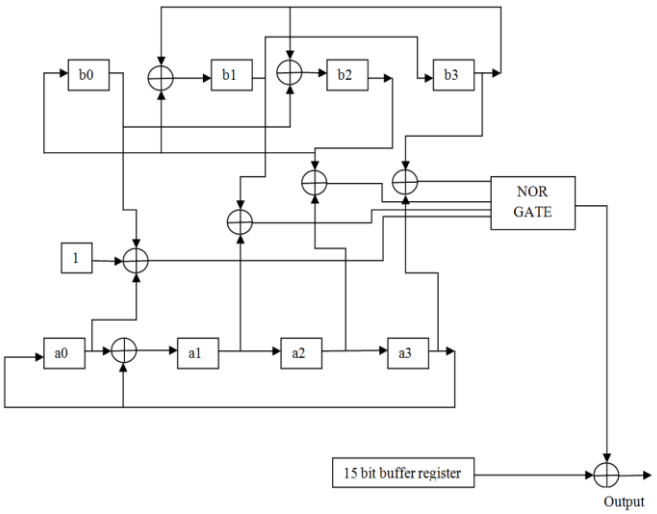

Fig. 12. Chine's search circuit for the double error correction (15, 7).

The third step is Correction of the errors. Using the error location information, Errors are corrected by simply flipping (converting 0 to 1 and 1 to 0) the data in the received 15 bit code word. Output of (15, 7) BCD decoder is the ASCII form of the corrected data.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

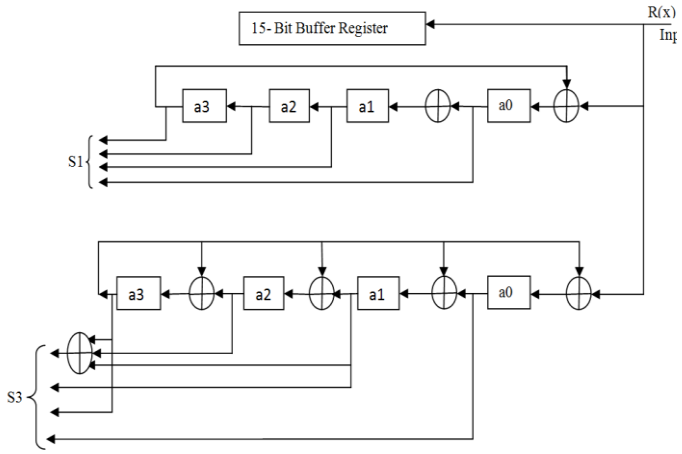

Fig -12: Syndrome circuit for (15, 7) BCH Decoder.

Table. 3. FPGA Resource Utilization.

Parameters	Used	Available	Utilization
Slice Registers	163	28,800	1%
Slice LUT's	256	28,800	1%
Block RAM/ FIFO	3	60	5%
Total Memory used (KB)	90	2160	4%
Total Power	0.48 Watts		

IV. CONCLUSION

In this paper, we implement Write Pattern Formatting algorithm (WPFA) of low complexity to improve the data retention reliability and power consumption of NAND flash based SSDs. If any data is being corrupted during read, process and write operation is taken care by BCH encoder and the decoder. The system has been simulated using Xilinx13.4 ISE simulator and functionality of WPFA and BCH encoder's and decoder's long with Multichannel NAND Flash memory are verified. Finally, hardware synthesized results over Spartan 3E demonstrate that the implementation complexity of the proposed scheme is much less than that of asymmetric coding and SPEA.

REFERENCES

[1] G. Dong, S. Li and T. Zhang, "Using Data Post-compensation and Pre-distortion to Tolerate Cell-to-Cell Interference in MLC NAND Flash Memory," IEEE Trans. Circuits Syst.I, Reg. Papers, vol. 57, no. 10, pp. 27182728, Oct 2010.

[2] S. Tanakamaru, C. Hung and K. Takeuchi, "Highly reliable and Low Power SSD Using Asymmetric Coding and Stripe Bitline-Pattern Elimination Programming," IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 8596, Jan 2012.

[3] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S.Unsal and K. Mai, "Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Life time," in IEEE 30th Int. Conf. on Computer Design (ICCD), pp. 94101, 2012.

[4] D. Hwan Lee and W. Sung, "Estimation of NAND flash memory threshold voltage distribution for optimum soft-decision error correction," IEEE Trans. Signal Process.,vol. 61, no. 2, pp. 440449, Jan 2013.

[5] Da-Chun Wu, AND Ming-Kao Hsu, "Authentication of Binary Document Images Based on Embedding the BCH Codes of Watermarks", Asian Journal of Health and Information Sciences, Vol. 1, No. 4, pp. 446-455, 2007.

[6] Laurencin Mihai Ionesco, Constantin Anton, "Hardware Implementation of BCH Error-Correcting Codes on a FPGA",International Journal of Intelligent Computing Research (IJICR), Volume 1, Issue 3, June 2010.

[7] P. Reviriego, C. Aggrades, J. A. Maestro, "Efficient errordetection in Double Error Correction BCH codes for memoryapplications", Microelectronics Reliability (2012) 1528–1530.

[8] Mahadevaswamy V P, Sunitha S L, B N Shobha, "Implementation of Fault Tolerant Method Using BCH Codeon FPGA", International Journal of Soft Computing and September 2012.

[9] S.H Park, S.H Ha, K Bang, "Design and Analysis of Flash Translation Layers for Multi-channel NAND Flash based Storage Devices", IEEE Transactions on Consumer Electronics, vol.55, n0.3 , pp. 1392-1400, August 2009.

[10] E. Gal and S. Toledo, "Algorithms and data structures forflash memories," ACM Computing Surveys, Vol. 37, No. 2, pp.138-163, Jun. 2005.

[11] Tang Lei, Zhou Xuan, Wu Yao, Li Jincheng, "Flash controller design for FPGA application", ICEEE 2010 International Conference , pp. 1-4, Nov. 2010.

[12] Chanik Park, PrakashTalawar, Daesik Won, Myung Jin Jung, Jung Been, Suksan Kim and Young joon Choi, "A high performance controller for NAND Flash-based solid state disk (NSSD)", IEEE 21st Non-Volatile Semiconductor Memory Workshop, pp. 17-20, Feb. 2006.

[13] Yu-Hsiang Kao and Juinn-Dar Huang, "High-Performance NAND Flash ControllerExploiting Parallel Out-of-Order Command Execution", 2010 IEEE 978-1-4244-5271-2/10.

[14] Tang Lei, Zhou Xuan, Wu Yao and Li Jincheng," Flash Controller Design for FPGA Application", 2010 IEEE 978-1-4244-7161-4/10.

[15] Yang Ou, Nong Xiao, Mingche Lai," A Scalable Multi-channel Parallel NAND Flash Memory Controller Architecture", 2011 Sixth Annual ChinaGrid Conference

[16] GongXin, Dai Zibin, Li Wei, Feng Lulu, "Design and Implementation of a NAND Flash Controller in SoC", 2011 IEEE 978-1-4577-1997-4/11.

[17] KoushelAgarwal, Vijay Kumar Magraiya, Anil Kishore Saxena, "Verification and simulation of New Designed NAND Flash Memory Controller", 2013 International Conference on Communication Systems and Network Technologies.

[18] Taeyeong Huh, Seolhee Lee, Yong Ho Song, "Scalable Approach for Flash Storage Controller Design", 2013 IEEE 978-1-4799-1142-4/13.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)