International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue14
November 2017

A Literature Survey on Ldpc Decoding of Nand Flash Memory by Using Array Dispersion

Ms. A . Kanchanachowdary & Mr. G . Mahendra
1 PG Student, Dept. Of VLSI and Embedded systems, SKR College Of Engineering & Technology, AP
2 Associate Professor, Dept. Of VLSI and Embedded systems, SKR College Of Engineering &Technology,AP.

ABSTRACT:

The reliability of data stored in an high-density Flash memory devices tends to drop off rapidly because of the reduced cell size and multilevel cell technology. The Soft-decision error correction algorithms make use of multiple-precision sensing for reading memory which can solve this problem. They require a very complex hardware for high-throughput decoding method. In this method, we present a rate-0.97 (68254, 65536) summarized Euclidean geometry low-density parity-check code and its VLSI operation for increased throughput NAND Flash memory system. The aim employ the normalized a posterior probability (APP)-based algorithm, serial schedule, and unrestricted update, which will lead to simple functional unit, halve decoding iterations, and compact power consumption. The pipelined equivalent structural design is adopted for high-throughput decodes, and memory-reduction techniques are employed to minimize the total chip size. The proposed decoder method is implemented in 0.13-µm CMOS tools, and the chip size and power consumption of the decoder are compared with those that of a BCH (Bose–Chaudhuri–Hocquenghem) decoding circuit performance compared with the error-correcting presentation and throughput.

Index Terms— Channel coding, low-density paritycheck (LDPC) codes, NAND flash memory, sequential scheduling

I. INTRODUCTION

NAND Flash memory is widely used in many mobile devices, such as cellular phones, digital cameras, and smart pads, because of high capacity, fast access speed, and low power consumption. In particular, solid-state drives (SSDs) for notebook computers have become popular as high density NAND Flash memory devices are available. NAND Flash memory stores the information by changing the threshold voltage of floating gate transistors. Most of today's high-density NAND Flash memory devices store multiple (usually two) bits in a cell, and this multilevel cell (MLC) technology significantly increases the bit-error rate (BER).Moreover, the feature size of NAND

Flash memory shrinks, the number of electrons in the floating gate of a transistor also decreases, and as a result, the memory is very prone to charge loss caused by long data retention. The cell-to-cell interference (CCI) also increasingly deteriorates the reliability of information stored at the floating gates. It is also well known that SSD applications usually demand high program and-erase cycles, which greatly affects the reliability of NAND Flash memory. NAND Flash memory devices have a spare region at each page, where parity bits for error correction can be stored. Hard-decision decoding algorithms, such as Hamming and BCH codes, have been widely used for NAND Flash memory error correction. However, as the process technology scales down continuously, more advanced error-correcting codes are needed to keep NAND Flash memory reliable. It is especially important to employ error-correcting algorithms that show good performance with a limited parity data ratio. Soft-decision error correcting methods that sense the threshold voltage of a memory cell in multiple-precision can increase the error-correcting performance because the reliability of stored information can also be utilized.

II. Existing system:

The parity-check matrix of our proposed LDPC code without masking is shown in Fig. 4. It consists of 30 × 300 submatrices with a submatrix size (z) of 63. There are six successive nonzero submatrices in each column block. Due to the second level array dispersion, the locations of nonzero submatrices are downward cyclic shift by one position between each column group. Based on the NMS-VSS decoding algorithm, the parity-check matrix is divided into 300 groups (G = 300). In other words, one group is equivalent to one column block. The proposed decoder has one VNU block and six CNU block processing units. At each cycle, the VNU and CNU blocks process the six successive nonzero submatrices in one column block. Let us define $h_{i,j}$ as the submatrix located at i^{th} row block and j^{th} column block for $0 \le i < 30$ and $0 \le j < 300$. B_k is denoted as k^{th} CNU block processing unit for $0 \le k < 6$, and mi is denoted as the messages belong to i^{th} row block.

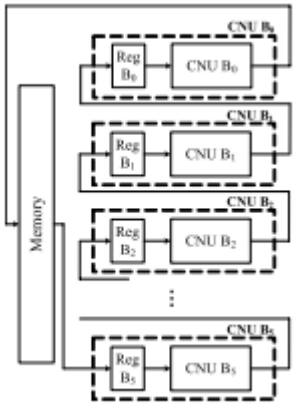

Fig 1: block diagram of the hybrid storage architecture

The registers in each block store the messages, which are going to be updated by VNUs and

CNUs at each cycle. After the update, the messages that will be processed in the next decoding cycle are stored in the registers again. This kind of messages is denoted as the immediate-use data. At the same time, the messages that are not needed in the next decoding cycle are stored in the SRAM for later use. This kind of messages is denoted as the non-immediate-use data. Fig. 6 shows more details of the access behaviour of the registers and SRAM. Consider at decoding cycle 0, B0, B5 will process h0, h5,0. When the invalid data are read from SRAM, it is replaced by predefined initial value (the infinite sign) and stored in the registers. After message update, the non-immediate-use messages will be stored in SRAM, while the immediate-use messages will be stored in the registers for next-cycle use. Let us recall the CNU from [18], there is a local sorter to get the minimum value from VNs located in one column group. Since our parity-check matrix is divided into 300 groups, there are 63 VNs in one column group. Each CNU out of 63 CNUs in one CNU block will receive the messages from one VN accordingly. In the other words, there is only one new message taken as an input to the sorter in each CNU. We can remove the local sorter from the CNU. It is not surprised that the CNU is now equivalent to the CNU with k = 2 in [25]. For each CNU, the registers store the min-and-index messages and 1-bit global sign bits. Meanwhile, the min-and-index message refers to the first and second minimum values and their indices. However, the data bandwidth of SRAM in the proposed architecture is only 1197 bits, which is reduced by 5/6. Moreover, about 24/30 block messages are stored in SRAM for better area efficiency compared with a conventional register-based architecture. The proposed hybrid architecture enjoys both high data bandwidth from register and better area efficiency from SRAM.

III. Proposed system:

Fig. 2 shows the overall decoder architecture and the details of one CNU block. There are one VNU block and six CNU blocks, where a VNU block includes 63 VNUs and each CNU block includes 63 sorters, 63 sign operation units, and two barrel shifters for messages and global sign bit. The data flows start from registers, then go through VNUs, CNUs, and barrel shifters. Finally, the data are written back to registers or SRAMs. As presented in Section II, the CV message includes the min-and-index message and global sign bit. $MCNU,B_k$ represents the 63 min-and-index messages stored in the registers of block B_k for $0 \leq k < 6$. It is sent to the sorters in block Bk. Then, the main message LC X,m,B_K is selected from $MCNU,B_k$ by the first/second min selector and sent to VNU.

After the VNU process, the absolute value of VC messages L XC,m,B_K is sent to sorters in

CNU block Bk. The sorters update the min-and-index messages from L XC,m,BK and MCNU, Bk. The new min-and-index messages M CNU,Bk after shifting by barrel shifters are then stored in the registers or SRAMs. The global sign bits are updated in the same manner as the min-and-index messages.

One may notice that the proposed parity-check matrix has maximum column degree of 5, but the hardware architecture adopts six CNU blocks. The reason of using six CNU blocks is to avoid serve global routing among CNU blocks. As shown in Fig. 2, the updated messages from each CNU block are stored in a corresponding register block. Despite the number of nonzero elements in each column group, six register blocks are required for storing messages of six successive row groups in order to implement the proposed hybrid storage architecture. Since the routing congestion is one of the critical challenges when implementing a large LDPC code, the hardware architecture adopts six CNU blocks to avoid additional multiplexers for selecting messages from and to the six register blocks before and after the update in CNU blocks.

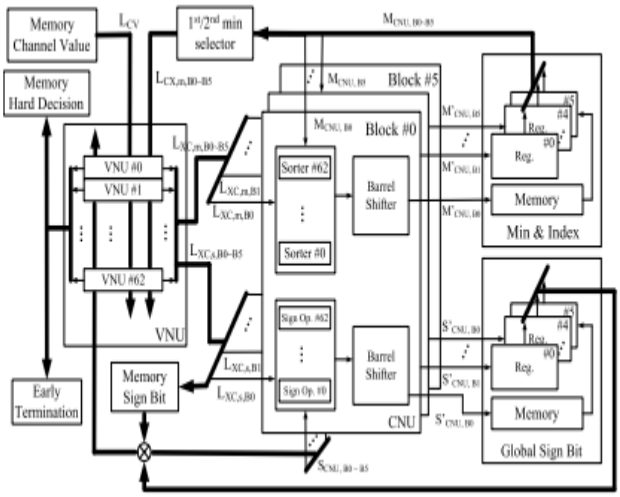

Fig 2: LDPC decoder architecture and details of one CNU block.

Since the decoder updates 63 VNs of one column block at each cycle, the data bandwidth of SRAM storing the channel values is (63 × 2 bit) = 126 bits. All the channel values are stored in a 300 × 126 single-port SRAM. In the beginning of the decoding process, the hard or soft-2-bit channel values are mapped to predefined log-likelihood ratio (LLR). The hard decision of VNU is written to a 300 × 63 single-port SRAM at each cycle. Finally, the two-port SRAMs storing min-andindex messages and global sign bits are replaced by singleport SRAMs to reduce silicon area by operating the SRAMs at double frequency. Note that the ECC protection for SRAM is not adopted in this paper. The NAND flash controller usually adopts the read retry technique [12] to reread and re-decode the data multiple times if correction fails. In addition, the failure of decoding due to soft error in

SRAM will not lead to a system crash. In general, the NAND flash controller will report a read failure to the OS system and the system will handle the failure as an exception case.

We implement the proposed decoder architecture on FPGA and investigate the error-floors of (18 900, 17 010) LDPC code with and without masking for comparison. The early termination technique is adopted and the maximum iteration is set to 20. The decoding throughput of the presented decoder implemented on Xilinx Virtex-5 LX330FPGA FPGA board is ~187 Mb/s under a clock frequency of 66 MHz. We also provide the FER and BER performances of the proposed LDPC codes over AWGN channel with x-axis of RBER. The error-floor of our designed LDPC code with column degree of 4 or 5 is not observed down to BER of 10−12. In contrast, the error-floor of LDPC code without masking is observed at BER of 10−10. The simulation results show that removing short cycles and trapping sets by masking matrix can efficiently suppress the error-floor.

IV. Simulation results:

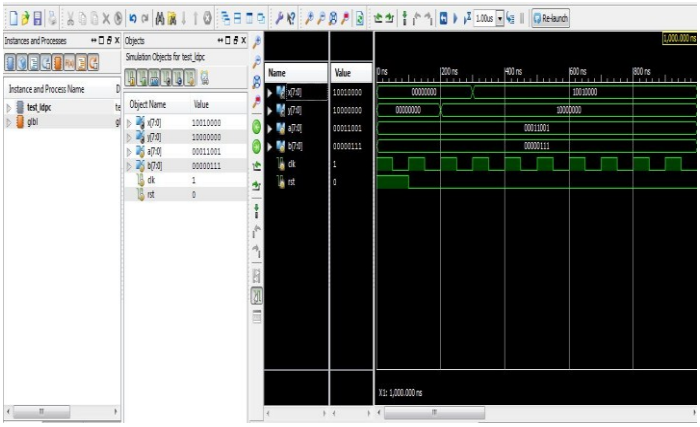

Fig 3: simulation result for the proposed system

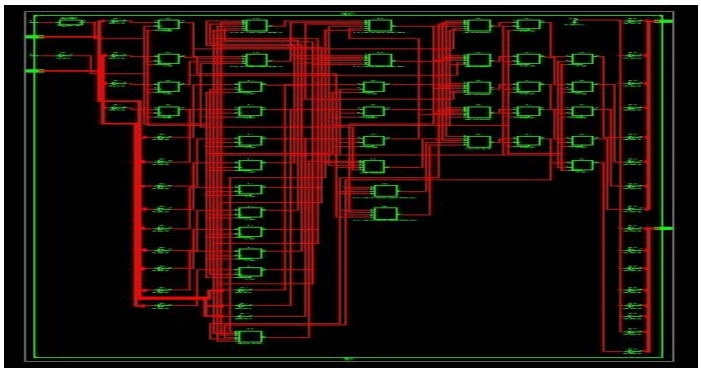

Fig 4: technology schematic for the proposed system

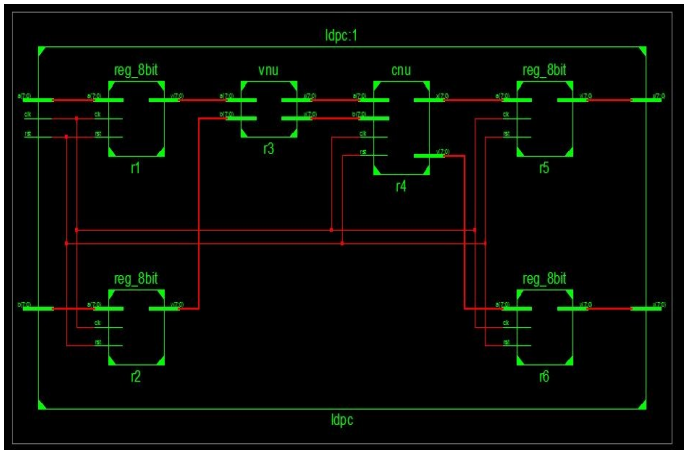

Fig 5: RTL of the proposed system

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	32	126800	0%
Number of Slice LUTs	16	63400	0%
Number of fully used LUT-FF pairs	16	32	50%
Number of bonded IOBs	34	210	16%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig 6: summary report of the proposed system.

V. CONCLUSION

We have demonstrated the proposed design methodology consisting of code construction, optimization, and hardware implementation. The presented two-step approach can construct LDPC codes not only satisfy long code length, high code-rate, good correcting capability, and low error floor, but also have hardware-friendly features, such as small submatrix size and diagonal-like structure of nonzero elements. The code optimization adopts masking to further improve the BER performance and suppress the error-floor. In hardware implementation, the small submatrix size efficiently solves the issues of large barrel shifter and worse routing congestion. Meanwhile, hybrid storage architecture is proposed based on the diagonal-like structure of nonzero elements. The implementation results show that the proposed decoder can achieve a throughput of 1.58 Gb/s with a gate count of 520k only. In addition, the proposed LDPC decoder meets the throughput requirements of both Toggle DDR 1.0 and ONFI 2.3 NAND interfaces.

REFERENCES

[1] Y. Li et al., "128 Gb 3 b/cell NAND flash memory in 19 nm technology with 18 Mb/s write rate and 400 Mb/s toggle mode," in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC), Feb. 2012, pp. 436–437.

[2] G. Naso et al., "A 128 Gb 3 b/cell NAND flash design using 20 nm planar-cell technology," in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC), Feb. 2013, pp. 218–219.

[3] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," Inf. Control, vol. 3, no. 1, pp. 68–79, 1960.

[4] A. Hocquenghem, "Codes correcterusd'erreurs," Chiffres, vol. 2, pp. 117–156, Sep. 1959.

[5] R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inf. Theory, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[6] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in Proc. 5th IMA Conf. Cryptogr. Coding, Oct. 1995, pp. 100–111.

Author's Profile

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue14
November 2017

Ms.A.KANCHANA CHOWDARY received B.Tech in Electronics and Communication Engineering from SKR College of Engineering, Nellore affiliated to the Jawaharlal Nehru technological university Anatapur in 2011, and pursing M. Tech in VLSI and Embedded systems from SKR College of Engineering affiliated to the Jawaharlal Nehru technological university Anantapur in 2015, respectively.

Mr.G.Mahendra Working as Associate Professor Department of ECE.

Qualification: M.Tech

Specialization: VLSI System DesignSKR College of Engineering & Technology

Email ID: Mahendra.goni@gmail.com