Byte Reconfigurable LDPC Codec Design for High Performance Error Correction

[1]Anumol Thomas, [2]Abhila R Krishna
[1] PG Student [VLSI & ES] [2] Assistant professor,
Department of ECE, TKM Institute of Technology, Kollam
[1] anuthomas180392@gmail.com [2] abhilaktvm@gmail.com

***Abstract*:-- Error correction provides reliable information free from errors. BCH codes are widely used to perform error correction. However, in order to improve the reliability of error correction BCH codes have been replaced by Low Density Parity Check (LDPC) codes. LDPC codes are special class of error correcting codes widely used in communication and memory systems, due to its Shannon limit approaching performance and favorable structure. A sub class of LDPC codes, called Quasi-Cyclic (QC) LDPC codes is used as the error correcting code due to their structured Parity Check Matrices (PCM). These codes are flexible in the sense of supporting wide range of code lengths and rates. This work deals with an efficient byte reconfigurable, high throughput QC-LDPC codec design. The codec is able to support multiple bits with the constraint that size of the sub-parity matrix be multiples of eight. Error detection is done using modified majority logic decoding. Majority logic decoding is preferred as they can correct large number of errors having even as well as odd number of bit flips. This method reduces the decoding time by detecting errors up to four bit-flips in first three iterations of decoding. If no error is detected decoding terminates without completing rest of the iterations, thereby reducing the average decoding time.**

***Index Terms*— Error Correction Codes (ECC), Forward Error Correction (FEC), Majority Logic (ML) decoding, Quasi-Cyclic LDPC codes.**

I. INTRODUCTION

As technology continues to scale down, memory has been increasingly relying on error-correction codes (ECCs) to ensure the overall data storage integrity. The aim of error protection technique is to provide reliable information which is free from errors. Coding scheme is implemented on systems with help of encoder and decoder. Encoder is used to secure the information and is transferred in the form of bits to the receiver. The receiver gets the digital data which is affected by errors from channel and is then passed to the decoder which retrieves the information. The principal of encoding and decoding is defined as adding or removing redundancies which are in the form of parity information. Parity information is defined as randomly generated sequence to secure the transmission systems. At the receiver redundancies detect as well as correct the errors in the information.

Error protection is accomplished in Forward Error Correction (FEC) coding scheme by implementing appropriate encoding and decoding techniques. FEC provides ability for the receiver to correct errors without requiring any reverse channel for retransmission of data. Therefore, it is applied in situations, where retransmission of information is expensive or not possible such as broadcasting to multiple receivers. As a result, FEC codes are applicable, where low bandwidth and latency are required.

Currently, BCH codes are widely used to perform ECC in memory. However, the advanced manufacturing technology, together with the use of multi-level cell (MLC)[1], deteriorates the reliability of memory. Recently, low-density parity-check (LDPC) codes with soft-decision are widely used as ECC to solve the reliability issues.

Compared to the BCH codes, LDPC codes show much superior performance with the same parity bits. Low-density parity-check (LDPC) codes[12] are special class of linear block codes widely used in communication and disk storage systems due to their Shanon limit approaching performance and favorable structure. The name comes from the characteristic of their parity-check matrix which contains less number of 1's when compared to the number of 0's. Main advantage of the code is that they provide a performance which is close to the capacity for a lot of different channels and linear time complex algorithms for decoding. Furthermore, are they most suitable for implementations that make heavy use of parallelism?

Based on the methods of construction LDPC codes are classified into two categories[2]. A LDPC code is called regular if the total number of 1s in each column(w_c) is constant and total number of 1s in each column is equal to the total number of 1s in each row i.e, $w_r = w_c$. If H is low density but the numbers of 1's in each row or column aren't constant then the code is called as an irregular LDPC code.

QC-LDPC which is a special class of regular LDPC codes is the ECC used here. These codes provide better encoding when compared to other LDPC codes. QC LDPC codes are widely used as the forward error correcting codes due to their structured parity check matrix. Each row of the parity check matrix is obtained by cyclic shifting of its previous row. Thus, only a single row needs to be stored in the memory which reduces the memory cost as well as complexity of the system. The structured parity-check matrix and the cyclical-shift property of circulant significantly simplify the hardware implementation of QC-LDPC codec compared to other non-structure LDPC codes. That is, QC codes can be

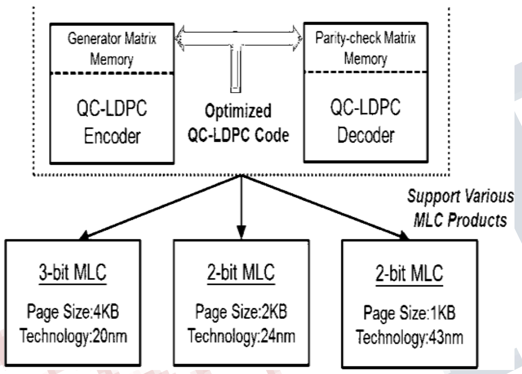


Fig.1 Byte Reconfigurable QC-LDPC codec.

Encoded with simple shift registers, based on their corresponding generator matrices. Therefore, in practical applications, they are strong competitors to the random codes, because of low error floors and simple encoding capability. These codes are also applicable in IC decoder implementations due to their cyclic symmetric structure, which results in simple regular wiring and modular structure.

In this system the objective is to develop a byte reconfigurable, cost effective, high throughput QC-LDPC codec by introducing some additional features to the existing QC-LDPC codec. Since quasi-cyclic LDPC codes significantly reduce the hardware complexity and is adopted in many communication systems our design supports QC-LDPC codes.

The main contributions of the system are as follows:-

1. A byte reconfigurable codec reduces redesign effort. In addition, because the codeword length of LDPC equals page size and page size of memory is multiples of byte, a reconfigurable LDPC codec is designed. The codec supports QC-LDPC codes of varying size. Hence, the design is named as byte-reconfigurable codec.
2. To reduce the memory cost of codec a shared-memory architecture is designed. In convention codec each sub-parity generator has separate memory bank which is accessed in a parallel and independent fashion, which increases the fixed memory cost of the system. In this system instead of separate private memories for each sub parity matrix, a single shared memory is used which reduces the fixed memory cost of the system.
3. A Majority Logic(ML) decoding circuitry is used as fault detector to accelerate the read operations. A modified version of ML decoder is designed for efficient identification of 'false negatives' as well as to decode upto four bit flips in three decoding cycles.

II. LITERATURE REVIEW

A. Conventional Encoder Design

The conventional encoder applies private memory architecture [10], where each sub-parity generator has a unique memory bank. Sub-parity generators access their memory bank in a parallel and independent fashion. The corresponding memory access strategy is shown in Figure 2. Because each sub-parity generator requires a private memory, many shallow-depth memories are contained in the encoder. The periphery circuits of a memory, such as row decoder, column decoder, and sense amplifier, contribute to

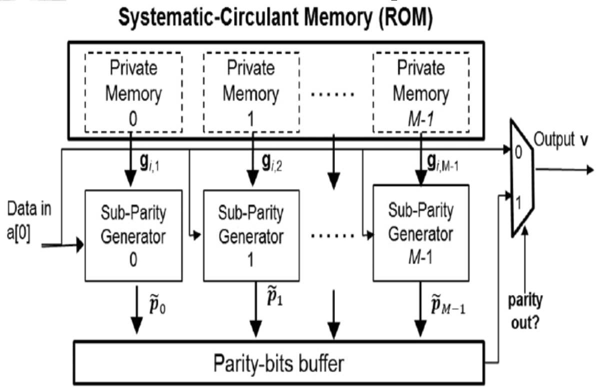


Fig.2 Conventional QC-LDPC encoder.

Fixed cost of the memory. Therefore, those small-sized memories increase the fixed cost of the systematic-circulant memory bank and leads to high memory cost..

Hardware architecture of QC-LDPC encoder[10] is shown in Figure 2. The encoder is composed of a circulant memory and parallel fully-serial sub-parity generators. The

$g_{i,j}$ is the first row of circulant and is stored in circulant memory. Besides, every fully-serial sub-parity generator to generate the j^{th} parity check section can be formed with a shift-register-adder-accumulator with feedback shift register circuit. The input sequence is serially shifted into the encoder, and the sub-parity generator computes parity bits in a fully-serial way. At the beginning of the first step, $g_{1,1}$ to $g_{1,M-1}$ are parallel loaded into the b-bit feedback shift-register of each sub-parity generator and the content of register A is set to zero. The inputs to feedback shift-register are multiplied by the input bit through the AND gates, and the outputs of AND gates are accumulated with the content of b-bit register by the XOR gates. Outputs of the XOR gates are stored in the b-bit registers for next row operation. The second rows of $G_{0,j}$ to $G_{0,M-1}$are generated by cyclically right shifting the $g_{1,1}$ to $g_{1,M-1}$, which are stored in feedback shift register. Then, the procedure proceeds as those of the first row. The computation is finished after performing the mentioned operation times to compute u_0G_0. Afterwards, the $g_{2,1}$ to $g_{2,M-1}$ are parallely loaded into the feedback shift-registers and it takes another b cycles to generate the partial sum, which is denoted as $u_0G_{0,j}$ +$u_0G_{0,j}$. At last, each parity bit is determined after (N-M) repetitive processing. Furthermore, all the parity bits, from P_0 to P_{M-1}, are generated by sub-parity generators. Figure.2 shows the architecture of conventional QC-LDPC encoder.

The major disadvantages of this codec include high fixed cost memory. Here, each sub-parity generator uses a private memory and thus many shallow-depth memories are contained in the encoder. The periphery circuits of a memory, such as row decoder, column decoder, and sense amplifier also contribute to fixed cost of the memory.

B. Majority Logic Decoding

The Majority Logic decoders[6] can correct multiple bit-flips based on the number of parity check equations. The decoder consists of a cyclic shift register, a majority gate and an XOR matrix. The input signal x which corresponds to wrong data corrupted by a soft error is initially loaded into the cyclic shift register. The XOR matrix uses the output values from cyclic shift registers to calculate the check sum equations.

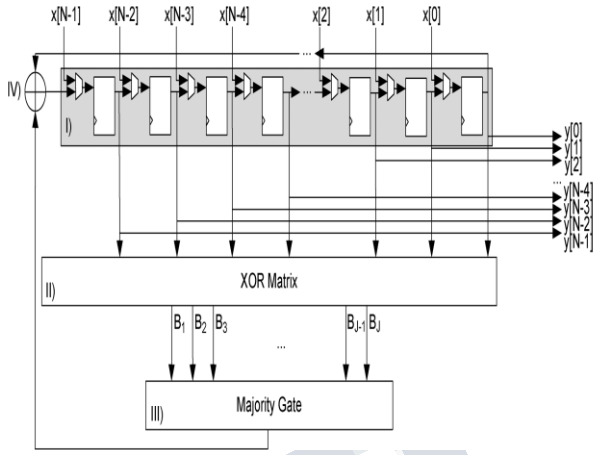


Fig..3 Schematic of an ML decoder.
(a) cyclic shift register. (b) XOR matrix. (c) Majority gate. (d) XOR for correction.

The resulting sums B_J are then given to majority gate for checking its correctness.

A faulty codeword is detected, if the number of 1's received in B_J is greater than 0's. In this case a signal would be triggererd to correct it. Otherwise, if the bit under decoding is correct it is forwarded directly to the output, thereby saving the correction cycles. In the next step, content of cyclic shift register is rotated and the above procedure is carried out. This process is repeated until all the N codeword bits are processed. Finally, a correct codeword is said to be decoded if the parity check sums are zero.

The main disadvantage of this algorithm is, it requires as many cycles as the number of input signal bits. For example, for a codeword having 73 bits, the number of decoding cycles required is 73, which is excessive in most of the applications. ML decoder performs error detection in a simple way using parity check sums. However, in case of multiple bit flips, the decoder may misbehave i.e., the decoder can correctly detect any number of odd bit flips by producing a '1' in the corresponding checksum. However, the parity check equation fails to detect the error in case of even number of bit-flips.

III. PROPOSED SYSTEM

A. Encoding Of QC-LDPC Codes

ECC starts with the construction of parity check matrix. From the parity check matrix H the generator matrix G can be constructed. If we put the sparse matrix H in the form: $H=[P^TI]$ (1)

Then the generator matrix can be calculated as:

$$G = [I|P] \quad (2)$$

The encoding of an (n,k) LDPC codes is identical to that of linear block codes by generator matrix, G. For an input message sequence u, there exists codeword v that satisfies

$$v = u.G \quad (3)$$

In case of QC-LDPC codes, the desired generator matrix can be shown in systematic-circulant (SC) form, as

$$\mathbf{G} = \left[\begin{array}{cccc|cccc} \mathbf{I} & \mathbf{O} & \cdots & \mathbf{O} & \mathbf{G}_{0,0} & \mathbf{G}_{0,1} & \cdots & \mathbf{G}_{0,M-1} \\ \mathbf{O} & \mathbf{I} & \cdots & \mathbf{O} & \mathbf{G}_{1,0} & \mathbf{G}_{1,1} & \cdots & \mathbf{G}_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{I} & \mathbf{G}_{N-M-1,0} & \mathbf{G}_{N-M-1,1} & \cdots & \mathbf{G}_{N-M-1,M-1} \end{array}\right] \quad (4)$$

In generator matrix G, I is a b*b identity matrix, O is a zero matrix, and $G_{i;j}$ is a circulant of size b*b. G consists of two parts, the left part I and the right part P. G in this form is known to be in systematic form, and the right part P of G is called P matrix that corresponds to the parity-check section of a codeword in systematic form. G is said to be in SC form, because its P matrix is an array of circulants. The SC form allows encoding a QC-LDPC code with shift registers. QC-LDPC encoding of parity part utilizes the cyclic shift property of circulants in P. The parity generator is composed of M sub-parity generators, the jth sub-parity generator calculates the jth section of parity bit as given in equation (5). Here u_i is b consecutive bits of ith section of u , and p_j is b consecutive bits of jth section of parity part p.

$$p_j = u_0G_{0,j} + u_1G_{1,j} + \dots + u_{N-M-1}G_{N-M-1,j} \quad (5)$$

B. *Design Of Byte Reconfigurable Encoder*

The entire byte-reconfigurable encoder architecture[7] is shown in Figure 3. The generator matrix is initially fed to the memory. Then the information sequence is serially encoded. The parity sections are formed simultaneously in parallel, and are finally shifted to the channel serially. The encoding circuit for the codeword can be obtained based on the generators of the circulants in the P matrix of G.

Let the information sequence of (t-c)b bits to be encoded be a = (a_1,a_2, ... $a_{(t-c)b}$). Divide this sequence into (t-c) equal length sections, a = (a1; a_2, ...$a_{(t-c)}$), where for the ith section a_i consists of b consecutive information bits, a = ($a_{(i-1)b+1}$; $a_{(i-1)b+2}$; ....$a_{(i)b}$). Then the codeword for the information sequence a is v = aG, which has the following systematic form:- v = (a; p1; p2; .....pc), pj = (pj,1; pj,2; ......pj,b) is a section of b parity-check bits. It follows from v = aG that,

$$p_j = a_1G_{1,j} + a_2G_{2,j} + \dots + a_{t-c}G_{t-c,j} \quad (6)$$

let $g_{i;j}^{(l)}$ be the lth cyclic shift generator of $g_{i;j}$ with $g_{i;j}^{(0)} = g_{i;j}^{(b)} = g_{i;j}$. Then,

$$a_iG_{i,j} = a_{(i-1)b+1}g_{i,j}^{(0)} + a_{(i-1)b+2}g_{i,j}^{(1)} + \dots + a_{(i)b}g_{i,j}^{(b)} \quad (7)$$

It follows from previous equations that the jth parity-check bit p_j can be computed as information sequence is shifted into the encoder in a step by step manner.

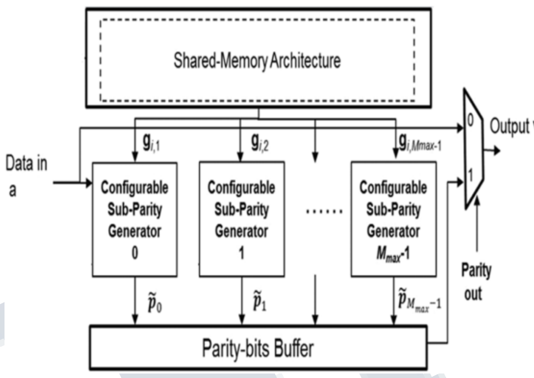

Shared-Memory Architecture
Data in
a
$g_{i,1}$
$g_{i,2}$
$g_{i,M_{max}-1}$
Configurable Sub-Parity Generator 0
Configurable Sub-Parity Generator 1
......
Configurable Sub-Parity Generator M_{max}-1
0
1
Output v
Parity out
$\tilde{p}_0$
$\tilde{p}_1$
$\tilde{p}_{M_{max}-1}$
Parity-bits Buffer

***Fig.4. Byte-Reconfigurable Encoder Design*.**

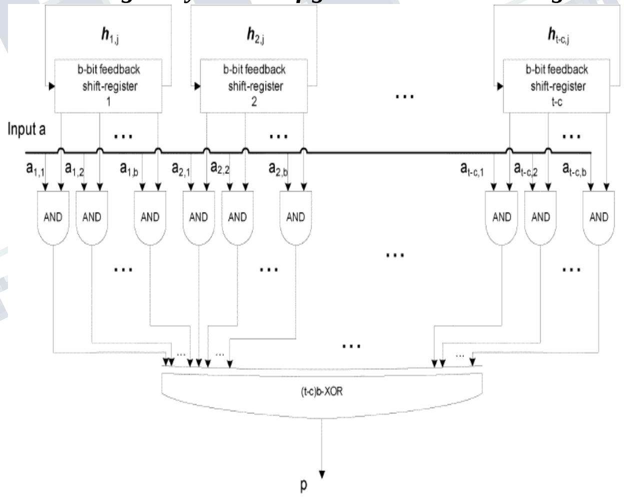


***Fig.5. Byte-Reconfigurable SRAA Encoder circuit*.**

At the kth step, the accumulated sum $s_{k;j} = a_1G_{1;j} + a_2G_{2;j} + \dots + a_kG_{k;j}$ is formed and stored in register. At the (k +1)th step, the partial sum $a_{k+1}G_{k+1;j}$ is computed from equation and added to $s_{k;j}$ to form the next accumulated sum $s_{k+1;j}$. At the end of the (t-c)th step, the accumulated sum $s_{t-c;j}$ gives the jth parity section pj in a register. At the (k +1)th step, the partial sum $a_{k+1Gk+1;j}$ is computed from Eq.(7) and added to $s_{k;j}$ to form the next accumulated sum $s_{k+1;j}$. At the end of the (t-c)th step, the accumulated sum $s_{t-c;j}$ gives the jth parity section p_j.

Based on encoding process the jth parity check section pj can be obtained by a shift- register -adder-accumulator (SRAA) circuit, as shown in Fig.5. Here, $h_{i;j}$

forms the first column of circulant $G_{i;j}$. For $h^{(l)}_{i;j}$ denote the lth downward cyclic shift of $h_{i;j}$; where $h^{(0)}_{i;j} = h^{(b)}_{i;j} = h_{i;j}$. Then the l^{th}, parity-check bit $p_{j;l}$ of the j^{th} parity section p_j is given by,

$$p_{j,l} = a_1 h_{i,j}^{(l-1)} + a_2 h_{2,j}^{(l-1)} + \dots + a_{t-c} h_{(t-c,j)}^{(l-1)} \qquad (8)$$

To obtain b parity bits of jth parity section pj , the columns $h_{i;j}$......$h_{t-c;j}$ are loaded parallely into the (t-c) feedback shift registers, and the first parity-check bit $p_{j;1}$ of p_j is thus formed.

C. Shared Memory Architecture For Systemic Circulant Memory

In LDPC codes for memory system the codeword length is usually multiple KBs. $g_{i;j}$ is stored in a systematic-circulant(SC) memory. The generator matrix G occupies a large proportion of encoder area. Since codeword length of LDPC codes is multiple of byte, the byte-reconfigure QC-LDPC encoder is designed. The architecture of the byte-reconfigurable encoder is shown in Figure 4. The conventional encoder contains many shallow depth memories inorder to implement seperate private memory architecture for each sub-parity generator. These shallow depth memories increase the fixed cost of systematic-circulant memory bank. Therefore, it is desired to use a deep-depth memory instead of many shallow-depth memories to store the generator matrix. Shared-memory architecture for the systematic-circulant memory bank is designed based on few deep-depth configurations. Sub-parity generators share a single memory bank, composed of one deep-depth memory. Since deep-depth memory reads out serially, only a single sub-parity generator can access memory bank output at a time. To eliminate data conflict among the sub-parity generators, serial memory access is also proposed for the shared-memory architecture. Sub-parity generators sequentially load the data from the systematic-circulant memory without conflicts.

D. Majority Logic Fault Decoder\ Detector (MLDD)

A modified form of Majority Logic Decoding[5] circuitry is used for fault detection in the codec. The decoding algorithm is almost similar to plain ML decoder but the performance is improved in exchange of some additional modules. The main contributions of the modified ML decoder includes:-

- ❖ Any error pattern effecting upto five bit in a codeword can be detected in three decoding cycles instead of N decoding cycles. This improves the performance of the system when compared to traditional majority logic decoders.

- ❖ Better handling of false negatives. In traditional parity check detector, odd number of errors would not create any problem. However, the parity check equation would not detect errors in case of even number of bit-flips. MLDD enables efficient handling of even as well as odd number of bit-flips.

The flow diagram of the MLDD algorithm is shown in figure 6. The decoding algorithm is similar to plain ML decoder[5]. The major difference is that, instead of decoding all codeword in N cycles the MLDD stops in third cycle. As shown in the flow diagram, codeword is determined to be error-free if the output of XOR matrix for all $\{B_j\}$ is "0" in first three cycles of decoding process. In this condition decoding process is terminated as a correct codeword is obtained thereby reducing the decoding time. On the other hand, if $\{B_j\}$ contains atleast a "1" in any of the three decoding cycles, the decoding process would continue to eliminate errors. A detailed schematic of the MLDD is shown in figure 7.

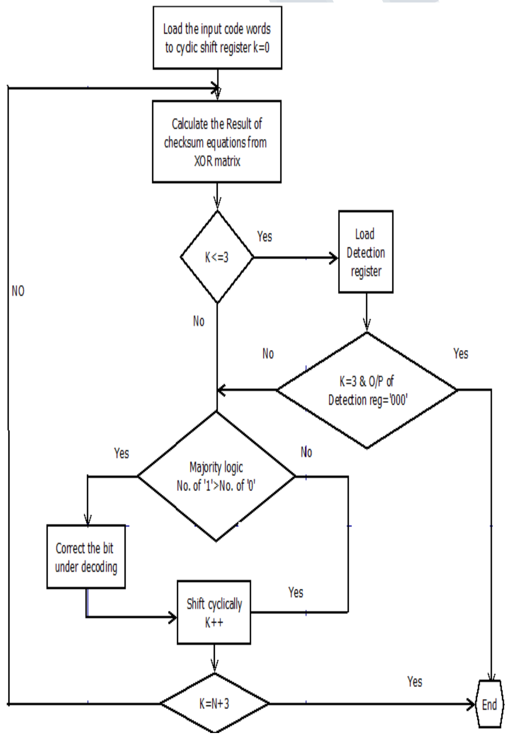

Load the input code words to cyclic shift register k=0
Calculate the Result of checksum equations from XOR matrix
K<=3
Yes
Load Detection register
NO
No
No
Yes
K=3 & O/P of Detection reg='000'
Yes
No
Majority logic No. of '1'>No. of '0'
Correct the bit under decoding
Shift cyclically K++
Yes
Yes
K=N+3
End

Fig.6. Flow diagram of the MLDD algorithm.

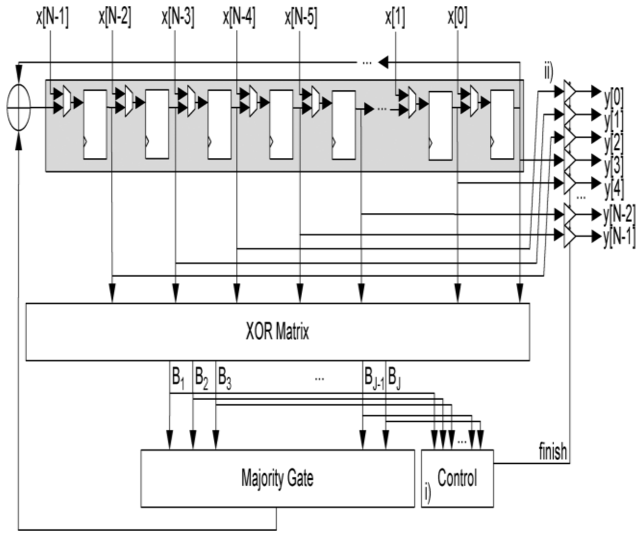


Fig.7 Schematic of the proposed MLDD. (a) Control unit. (b) Output tristate buffers.

The decoder consists of a n-tap shift register, XOR array, control unit, tristate buffer and Majority gate. XOR Matrix is provided to calculate the orthogonal parity check sums and a majority gate for deciding if the current bit under decoding needs to be inverted. The output tristate buffers are always in high impedance unless the control unit sends the finish signal so that the current values of the shift register are forwarded to the output. Majority gate using Sorting Networks decides if the current bit undergoing decoding process needs to be inverted.

The control unit manages the detection process by triggering a finish flag when no errors are detected after the third cycle. It uses a counter which counts up to three, which distinguishes the first three iterations of the ML decoding. In the first three iterations, the control unit evaluates $\{B_j\}$ by combining them with the first OR function. This value is then fed to a three-stage shift register, which holds the results of the last three decoding cycles. In the third cycle, the second OR gate evaluates the detection register content. If the result is “0” the FSM sends finish signal which indicates that the codeword obtained is error free and thus the process is terminated. Consider the other case, if result is “1,” the ML decoding processes till the last cycle.

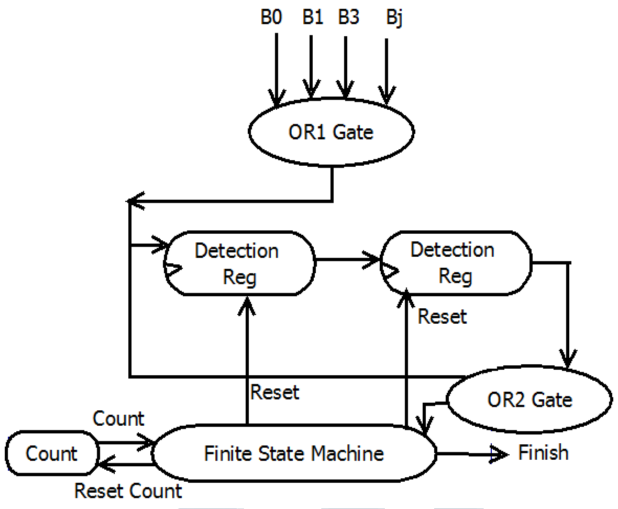


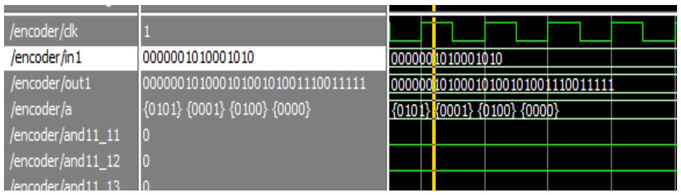
Fig.8 Schematic of Control unit.

IV. EXPERIMENTAL RESULTS

The design flow is modeled using VHDL in Xilinx ISE Design Suite 13.1 and the simulation of the Byte Reconfigurable Codec is performed using ModelSim SE 6.3f to verify the functionality of the design.

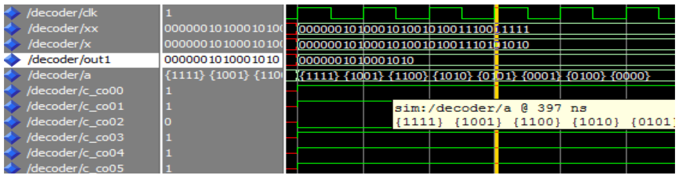
A. Simulation of Byte-Reconfigurable Codec :16bit information sequence

Simulation result of Byte-Reconfigurable encoder with 16 bit input data(in_1) in shown in Fig.9 and *out1* forms the corresponding 32 bit codeword. Simulation result of decoder is shown in Fig.10, where *XX* is the decoder input i.e, encoder output and *X* is the corrupted codeword. *out1* forms the corresponding decoded output i.e, decoder retrieves the 16 bit input message sequence from the 32 bit codeword.



```
/encoder/clk        1
/encoder/in1        0000001010001010                    0000001010001010
/encoder/out1       00000010100010100101001110011111    00000010100010100101001110011111
/encoder/a          {0101} {0001} {0100} {0000}         {0101} {0001} {0100} {0000}
/encoder/and11_11   0
/encoder/and11_12   0
/encoder/and11_13   0
```

Fig.9 Simulation of Byte-Reconfigurable Encoder(16 Bit).



```
/decoder/clk        1
/decoder/xx         000000101000101001   00000010100010100101001110011111
/decoder/x          000000101000101001   00000010100010100101001110101010
/decoder/out1       0000001010001010     0000001010001010
/decoder/a          {1111} {1001} {110   {1111} {1001} {1100} {1010} {0101} {0001} {0100} {0000}
/decoder/c_co00     1
/decoder/c_co01     1                    sim:/decoder/a @ 397 ns
/decoder/c_co02     0                    {1111} {1001} {1100} {1010} {0101
/decoder/c_co03     1
/decoder/c_co04     1
/decoder/c_co05     1
```

Fig.10 Simulation of Decoder.

B. Simulation of Byte-Reconfigurable Codec :32bit information sequence

Simulation result of encoder with 8 bit information bit is shown in Fig.11 where 8 bit information sequence is decoded to 16 bit codeword which is retrieved to original information sequence by the decoder

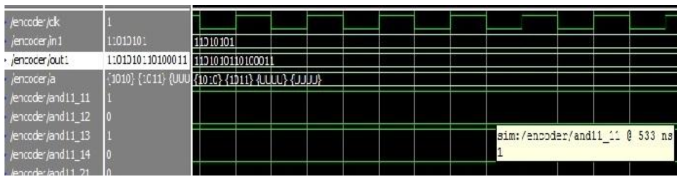


Fig.11. Simulation of Byte-Reconfigurable Encoder(32 Bit).

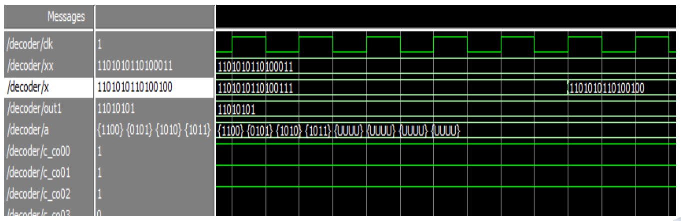


Fig.12. Simulation of Decoder.

V. CONCLUSION

A special class of LDPC code called QC-LDPC code is used as the error correcting code. In this work a byte reconfigurable QC-LDPC encoder and decoder which can handle multiple bits with the constraint the size of sub-parity generators be multiples of eight is designed and simulated. The byte reconfigurable codec reduces redesign effort of the system. The shared memory architecture is developed in the encoder as well as in the decoder sections to save memory cost. Modified majority logic decoding is used for efficient handling of even as well as odd number of bit flips where upto 4 bit-flips can be decoded in 3 clock cycles thereby reducing the decoding time. The simulation of codec is done in VHDL using Xilinx ISE Design Suite 13.2

REFERENCES

[1] Yu-Min Lin, Huai-Ting Li, Ming-Han Chung, and An-Yeu (Andy) Wu,"Byte-Reconfigurable LDPC Codec Design With Application to High-Performance ECC of NAND Flash Memory Systems", *IEEE Trans. Circuits Syst.* I: REGULAR PAPERSVOL. 62, NO. 7, July 2015

[2] Y. Kou, S. Lin, and M. P. C. Fossorier, Low density parity-check codes based on finite geometries: a rediscovery and new results, *Inform. Theory, via IEEE*, May 2015.

[3] Y. S. Park, Y. Tao, and Z. Zhang, A fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating, *IEEE J. Solid-State Circuits*, vol. 50, no. 2, pp.464475, Feb. 2015.

[4] J. Li, K. Zhao, J. Ma, and T. Zhang, Realizing unequal error correction for nand flash memory at minimal read latency overhead, *IEEE Trans. Circuits Syst. II*, Exp.Briefs, vol. 61, no. 5, pp. 354358, May 2014.

[5] Shih-Fu Liu, Pedro Reviriego and Juan Antonio Maestro Efficient Majority Logic Fault Detection With Difference-Set Codes for Memory Applications, *IEEE Trans On Vlsi Systems*, Vol. 20, No.1, Jan 2012

[6] Y. H. Chien, M. K. Ku, and J. B. Liu, Low-complexity iteration control algorithm for multi-rate partially parallel layered LDPC decoders, *IET Electron. Lett.*, vol. 48, no. 22, pp. 14061407, Oct. 2012.

[7] S. Kim, G. E. Sobelman, and H. Lee, A reduced complexity architecture for LDPC layered decoding schemes, *IEEE Trans. Very Large Scale Integr*. (VLSI) Syst., vol. 19, no. 6, pp. 10991103, Jun. 2011.

[8] X. Y. Shih, C. Z. Zhan, C. H. Lin, and A. Y.Wu, An 8.29 52mW multi-mode LDPC decoder design for mobile WiMAX system in 0.13 CMOS process, *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 672683, Mar. 2008

[9] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit error rate in NAND flash memories," in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2008, pp. 9–19.

[10] Z. W. Li, L. Chen, L.-Q. Zeng S. Lin, andW. H. Fong, Efficient encoding of quasi-cyclic low-density parity-check codes, IEEE Trans. Commun., vol. 54, no. 1, pp. 7181, Jan. 2006.

[11] M. M. Mansour, A turbo-decoding message-passing algorithm for sparse paritycheck matrix codes, *IEEE Trans. Signal Process*., vol.54, pp. 43764392, Nov. 2006

[12] M.C. Davey and D.J.C. MacKay, Low-density parity check codes over GF(q), *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165-167, Jun. 1998.