PERFORMANCE COMPARISON OF EG-LDPC CODES WITH MAXIMUM LIKELIHOOD ALGORITHM OVER NON-BINARY LDPC CODES

M.Sakthivel[1],M.Karthick Raja[2] ,KR.Ragupathy[2] and K.Sathis Kumar[2]

[1]Assistant Professor, Department of Electronics and Communication Engineering,
Velammal College of Engineering and Technology,Viraganoor,
Madurai, Tamilnadu, India
[2]UG Students,Department of Electronics and Communication Engineering,
Velammal College of Engineering and Technology,Viraganoor,
Madurai, Tamilnadu, India

ABSTRACT

Error correcting coding has become one essential part in nearly all the modern data transmission and storage systems. Low density parity check (LDPC) codes are a class of linear block code has the superior performance closer to the Shannon's limit. In this paper two error correcting codes from the family of LDPC codes specifically Euclidean Geometry Low Density Parity Check (EG-LDPC) codes and Non-binary low density parity check (NB-LDPC) codes are compared in terms of power consumption, number of iterations and other parameters. For better performance of EG-LDPC codes, Maximum Likelihood (ML) Algorithm was proposed. NB-LDPC codes can provide better error correcting performance with an average of 10 to 30 iterations but has high decoding complexity which can be improve by EG-LDPC codes with ML algorithm having only three iterations for detecting and correcting errors. One step majority logic decodable (MLD) codes is a subclass of EG-LDPC codes are used to avoid high decoding complexity. The power Consumed by NB-LDPC codes is 2.729W whereas the power consumed by EG-LDPC codes with ML algorithm is 1.148W.

KEYWORDS

Euclidean Geometry - Low Density Parity Check code, Maximum Likelihood Algorithm, Non Binary - Low Density Parity Check Code.

1. INTRODUCTION

Memory is a basic resource in every digital systems which should be protected from all kinds of faults for reliable performance, but nowadays, single event upsets (SEU) altering these memories by changing its state, which is caused by ions or electro-magnetic radiations. SEU is often referred to as a soft error which occurs when a radiation event causes enough charge disturbances to reverse or flip the data state of a memory cell. Also, soft errors change the logical value of memory cells without damaging the circuit. So, errors in a word stored in memory must be detected and corrected for reliable communications. The error detection and correction can be done in many ways with various error correcting codes. This paper mainly focused on family of Low Density Parity check codes namely NB-LDPC codes and EG-LDPC codes in order compare their performances. Non-binary low-density parity-check (NB-LDPC) codes can achieve better error-correcting performance than their binary counterparts when the code length is moderate at

the cost of higher decoding complexity. The high complexity is mainly caused by the complicated computations in the check node processing and the large memory requirement. NB-LDPC codes are defined by a sparse parity check matrix in a finite field or Galois field GF (q). This definition is for the purpose of improving Binary LDPC in small and moderate code words. They also provide the maximum transmission rate through channel which is very close to Shannon's limit. In nineties NB-LDPC codes were decoded with Belief propagation (BP) but they have a computational complexity dominated by O (q^2). But, due to Belief Propagation, it can provide better performance. They also provide better coding gain, lower error floor than binary LDPC. We need to note that the decoder complexity will increase, if we increase the value of q, which dominates the computational complexity more than 16. So, for the purpose of reducing it decoding complexity, Extended Min-Sum (EMS) algorithm is used. This EMS algorithm is based on the Min-Sum algorithm which is used for binary LDPC codes. The Main aim of this EMS decoder is to extend the principle of truncation which is used in NB-LDPC. Also an NB-LDPC code is formed by grouping of bits to symbols using GF elements. The factor graph of an NB-LDPC code has fewer edges compared to an equivalent binary LDPC code, which is leading to simpler wiring in a decoder implementation. Each edge in an NB-LDPC factor graph carries a vector of messages, thus the global interconnects can be bundled for higher area efficiency. However, GF processing involves a complex check node (CN) and variable node (VN): CN runs a forward-backward algorithm to consider possible.

Although Non Binary-LDPC code has an advantage than binary LDPC code, it is not much efficient when compared to EG-LDPC with ML algorithm. Because the main feature of EG-LDPC codes is one step majority logic decoding. With the help of this logic, it can avoid high decoding complexity. But it is absolutely not possible in Non Binary LDPC codes, where the decoding complexity can increase if we increase the value of q. Iterations also plays a major role in EG-LDPC codes with ML algorithm where nearly 10 error bits can be corrected with in three iterations, which is not possible in case of Non Binary LDPC. So, EG-LDPC codes have better performance in detecting and correcting errors with less power consumption than the NB-LDPC codes.

2. EXISTING METHODOLOGY

The Non-Binary LDPC code is defined by a sparse parity check matrix, which is denoted by H. The Matrix component belongs to Galois field GF (q). The regular LDPC's H matrix has constant column weight and constant row weight. The NB-LDPC codes can be decoded by iterative message passing on a factor graph. This matrix may also represented by a factor graph and each column is mapped to a variable node and each row is mapped to a check node. Here, the edge connects variable node V_j and check node C_i, if H (i, j) ≠0. A large number of algorithms have been proposed for this code with different error-correcting performance and implementation complexity. Here, we discussed some algorithms for decoding NB-LDPC codes. The EMS algorithm, which is used here in NB-LDPC codes, offers a good tradeoff. Its complexity becomes even lower by truncating EMS algorithm. Regarding its iteration, this NB-LDPC's code gain depends on the limit on decoding iterations. Increasing itfrom 10 to 30 improves the SNR by 0.15dB at a FER of 10-6, but the energy efficiency worsens. Here, we can notice that the majority of VNs converge very quickly before reaching the iteration limit. In addition, VNs and CNs are dominated by sequential circuits that consume significant clock switching power. So we design VN to detect its convergence and gate its clock to save power. The Convergence detection is based on two criteria: producing the minimum number of iterations and VN having maintained the same decision for the last consecutive iterations. The Convergence detection can be done at each VN level, which permits a fine-grained dynamic clock gating.

2.1Decoding algorithm of NB-LDPC codes:

Here, we can see about the decoding algorithms available for decoding NB-LDPC codes. There are five algorithms namely Belief Propagation Algorithm, EMS algorithm, simplified min-sum algorithm, Min-max algorithm and truncated EMS algorithm.

2.1.1 Belief-propagation Algorithm:

LDPC codes can be decoded by the belief propagation (BP) algorithm. In the belief propagation for non-binary codes, the complexity of check node processing dominates. A forward-backward scheme was proposed for BP in to decompose the check node processing into iterative steps. At the cost of larger memory, the number of computations can be significantly reduced.

2.1.2 Extended Min-sum (EMS) Algorithm:

Extended Min-Sum Algorithm is an approximation of the Belief Propagation in the log domain. It requires only addition in the check node processing. EMS algorithm can lead to coding gain loss of less than 0.1db compared to Belief Propagation. This can be done with proper scaling. In order to reduce the latency of computing the minimum of sums, the messages need to be kept sorted. Parallel sorters that are capable of inserting a value into a sorted vector of length *nm* in one clock cycle may lead to large area requirement, especially when multiple check or variable node units need to be implemented.

Dataflow of EMS algorithm:

For each received symbol of NB-LDPC codes, the corresponding transmitted symbol can be any of the *GF*(*q*) elements. The log-domain belief propagation can be approximated by the EMS and Min-max algorithms with small performance degradation. In the EMS, the reliability vector for a variable *z* consists of *q* log-likelihood ratios (LLRs): $L(\alpha) = \log(P(z = \beta)/P(z = \alpha))$, where α is an element of *GF*(*q*) and β is the most likely symbol for *z*. Using this definition, all LLRs are non-negative and they can be used as metrics to measure how far away the symbols are from the most likely symbol. The smaller the $L(\alpha)$, the more likely that the variable $z = \alpha$.

2.1.3 Simplified Min-sum algorithm:

The Simplified Min-sum algorithm relaxes the constraint of the check node processing. From the definition of the set $L(m|an = \beta)$, the LLRs from distinct variable-to-check (v-to-c) message vectors are added up to compute $vm,n(\beta)$ in the EMS algorithm. In the SMSA, a relaxation is made such that multiple LLRs from the same v-to-c vector can be included in the addition.

2.1.4 Min-max Algorithm:

The complexity of the check node processing is further reduced in the Min-max algorithm with slightly lower coding gain. In this algorithm, ’ max’ instead of ’ sum’ is carried out in the check node processing. In the Min-max check node processing architecture is proposed by keeping all *q* messages, and hence also suffers from large memory requirement when the order of the finite field is not small. In addition, this architecture has very long latency since it computes all possible combinations of field elements from each message vector. A parallel Min-max check node processing architecture was developed in. Although it can achieve high speed, it requires large area due to the high level of parallel processing. This architecture also becomes less 15 efficient when *q* is larger or the code rate is lower.

2.1.5 Truncated EMS Algorithm:

The EMS algorithm offers a good tradeoff. It achieves a performance close to the original BP algorithm and its complexity is relatively low. The truncated EMS algorithm achieves an even lower complexity and demonstrates great potential for practical adoption.

For completeness, we briefly introduce the five steps of the truncated EMS algorithm: (1) variable nodes are initialized with prior log-likelihood ratios (LLR): one LLR is associated with one of the q GF elements. The largest nm ($nm < q$) LLRs along with the corresponding GF element (or index) are stored in vector L and βL respectively in descending order; (2) variable-to check (v-c) messages are permuted based on the H matrix and sent to the check nodes. In the first iteration, the priors are used as the v-c messages; (3) for each adjacent variable node vj, check node ci computes the check-to-variable (c-v) message $\{V_{ij}[k]\}$, $k \in \{0, \ldots, nm - 1\}$, that the parity-check equation is satisfied if $vj = \beta V_{ij}[k]$ (where $\beta V_{ij}[k]$ is a vector containing GF elements or indices). The computation is done using a forward-backward recursion. Note that only the nm highest probabilities are computed and stored; (4) the c-v messages are inverse permuted before being sent to the variable nodes; (5) each variable node vj is updated with messages from the adjacent check nodes. A v-c message $\{U_{ji}[k]\}$, $k \in \{0, \ldots, nm - 1\}$, is computed for each adjacent check node ci, based on the prior L and all received c-v messages except from check node ci. The procedure repeats itself from step (2).

3. PROPOSED METHODOLOGY

The disadvantages in the existing methodology can be overcomes by implementing Maximum Likelihood algorithm with EG-LDPC codes in majority logic decoding process. In our proposed system, we introduced the concepts of ML algorithm along with EG-LDPC codes which will increase the performance by detecting and correcting up to ten bit errors in first three iterations which is not possible in the case of existing system.This provides controlling mechanism when the bit from word was extracted. High performance rate can be achieved because of this error bits can be reduced. Our proposed technique is done using VHDL programming language and simulated using Xilinx 14.2 software. Fig. 1 shows the block diagram of our proposed method in which the entire process is splitting into five modules namely Arrange bits, Check parity, Detect error, Correct error, Recover output.

3.1 ML Algorithm:

ML algorithm generates four signals from the bits of parity checker output.
Operation in signal 1:

- Condition matching – performs AND, NAND operation.
- Bit Combining – performs OR operation.
- Performs operation in D flip flop and mux to generate bit for further operation.

Operation in signal 2:

- Condition matching – performs OR, NOR followed by AND,NAND operation.
- Bit Combining – performs OR operation.
- Performs operation in D flip flop and mux to generate bit for further operation.

Operation in signal 3:

- It performs XOR operation.

Operation in signal 4:

- It performs counter operation.

In each operation, single bit is generated which is fed as input into Majority Logic decoder.

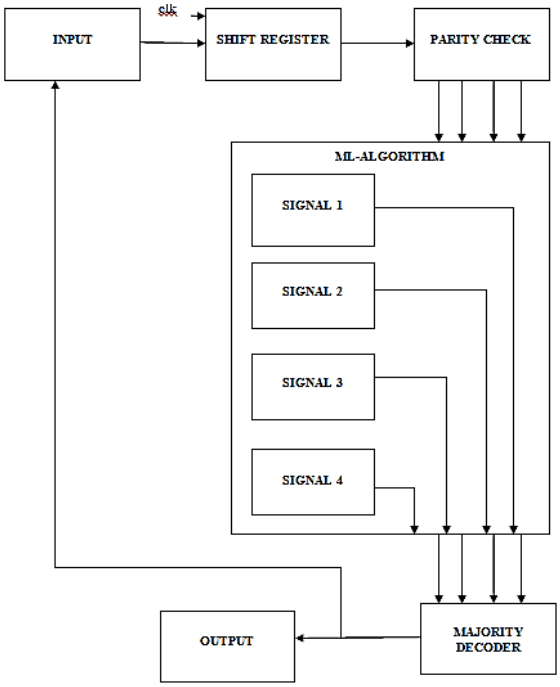

Figure1. Block Diagram of proposed method

3.1.1 Properties of Maximum likelihood method:

1. For large data samples (large n) the likelihood function, L, approaches a Gaussian distribution.
2. Maximum likelihood estimates are usually consistent. For large n the estimates converge to the true value of the parameters we wish to determine.
3. Maximum likelihood estimates are usually unbiased. For all sample sizes the parameter of interest is calculated correctly.
4. Maximum likelihood estimate is efficient: the estimate has the smallest variance.
5. Maximum likelihood estimate is sufficient.
6. The solution from MLM is unique.
7.

3.2 Modules and its Description:

1) Arrange Bits:
In this module, we arrange the words which are need to be error checking. These bits are arranged in bit by bit to shift register.

2) Check Parity:
In this module, we find the parity bit as a word to find error bits. There are N/4 numbers of Xor gates to get parity bit from shift register. These parity bits are consider as a parity word and fed this to majority logic circuits.

3) Detect Error:
In this module, we check the error bit from the parity word. For detecting and decodingmore number of error bits maximum likelihood algorithm is employed.Then, after this decoding we use MLD algorithm.

4) Correct error:

In this module, we correct the error bit by checking last bit from shift register and bit from majority logic circuit. Here, there was an error correction stage. In this error correction stage, we use XOR gate that changes error bit if there was change in bit. This compare bits by taking last bit from shift register and last bit from majority logic circuit and finally correct error bit.

5) Recover Output:

In this module, we get error free output from shift register. In this, if there was no any error in the word, then it stop cycle checking and goes to next word. Since the error correction was done for single bit, we can correct error bit if there was millions of error word.

3.3 Majority Logic Decoding (MLD):

Majority-logic decoding is a simple and effective scheme for decoding certain classes of block codes, especially for decoding certain classes of cyclic codes. The first majority-logic decoding algorithm was devised in 1954 by Reed for the class of RM codes. Most majority-logic decodable codes found so far are cyclic codes. Important cyclic codes of this category are codes constructed based on finite geometries, namely, Euclidean and projective geometries. These codes are called finite geometry codes and they contain punctured RM codes in cyclic form as a subclass. Majority logic decoding is the relatively efficient with low complexity error-correcting technique. MLD is based on the number of parity check equations which are orthogonal to each other, so that, at each iteration, each code word bit only participates in one parity check equation, except the very first bit which contributes to all equations. For this reason, the majority result of these parity check equations decide. Memory is processed by which information is encoded, stored and retrieved. While retrieving the information which is encoded should be uncorrupted. So it is very important to protect memory against error. A generic schematic of a memory system is depicted in Figure 2 for the usage of an ML decoder. Initially, the data words are encoded and then stored in the memory. When the memory is read, the code word is then fed through the ML decoder before sent to the output for further processing. In this decoding process, the data word is corrected from all bit-flips that it might have suffered while being stored in the memory.

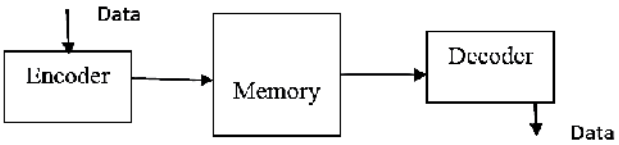

Figure 2.Simple memory system schematic

There are two ways for implementing this type of decoder. The first one is called the Type-I ML decoder, which determines, upon XOR combinations of the syndrome, where bits need to be corrected. The second one is the Type-II ML decoder that calculates directly out of the code word bits the information of correctness of the current bit under decoding. Both are quite similar but when it comes to implementation, the Type-II uses less area, as it does not calculate the syndrome as an intermediate step.

There are three kinds of Type-II ML decoder present for decoding process. They are

- Plain ML Decoder
- Plain MLD with Syndrome Fault Detector (SFB)
- Majority Logic Decoder/Detector

The ML decoder is a simple and powerful decoder, capable of correcting multiple random bit-flips depending on the numbers of parity check equations. It consists of four parts: A cyclic shift

register, XOR matrix, Majority gate and XOR for correcting the code word bit under decoding. In the next step, the content of the registers are rotated and the above procedure is repeated until all code word bits have been processed. Finally, the parity check sums should be zero if the code word has been correctly decoded.

Syndrome vector method overcomes the demerits of Majority Logic Decoder (MLD) method. The faulty code word is decoded, by adding the fault detector which calculates the syndrome value. This will not affect the performances of the system because most of the code word is error-free. The main drawback of this system increases the complexity to the design. Based on parity check equation, the XOR matrix calculates the syndrome value. This increases the complexity of the syndrome value vector based on the size of the code word. An error in the code word is identified when the syndrome vector value is '1', then the ML decoder is used to correct the wrong code word. Otherwise it forwards the code word to the output, without correcting cycles. In this method, the performance of the system is improved, but additional module which increases the complexity to the design. Further, it increases the power complexity and reduces the performances of the system. It will increase the power consumption. Syndrome vector is oldest technology, which is used to detect the error in the code word. Syndrome decoder is linear decoder. Hamming code is one of the examples of syndrome decoder.

The ML detector/decoder (MLDD) has been implemented using the Euclidean Geometry LDPC. The EG-LDPC are based on the structure of Euclidean geometries, among EG-LDPC codes there is an subclass of codes that is one step majority logic decodable (MLD) This method is very practical to generate and check all possible error combinations for codes with small words and affected by a small number of bit flips. When the size of code and the number of bit flip increases, it is difficult to exhaustively test all possible combinations. Therefore the simulations are done in two ways, the error combinations are exhaustively checked when it is feasible and in the rest of the causes the combinations are checked randomly. Since it is convenient to first describe the chosen design and also for simplicity, let us assume that the hypothesis is true, that only three cycles are needed to detect all errors affecting up to four bits in EG LDPC Codes.

4. COMPARISON

The power consumed by the components used for the construction of NB-LDPC codes and EG-LDPC codes with ML algorithm was simulated using Xilinx Power Estimator series7 XPE 2013. The power is calculated by loading number of flip-flops and slice LUT registers used. The comparison of Junction Temperature VS power and power consumed by elements for NB-LDPC and EG-LDPC with ML algorithm is shown in figure 3 to 6 respectively.

Table 1 Power analysis Summary

PARAMETERS	NB-LDPC codes	EG-LDPC codes with ML algorithm
Total Power	2.729W	1.148W
Iterations	15 to 30	3
Junction temperature	28.1°C	37.1°C
Thermal Margin	71.9°C, 62.2W	62.9°C, 5.7W

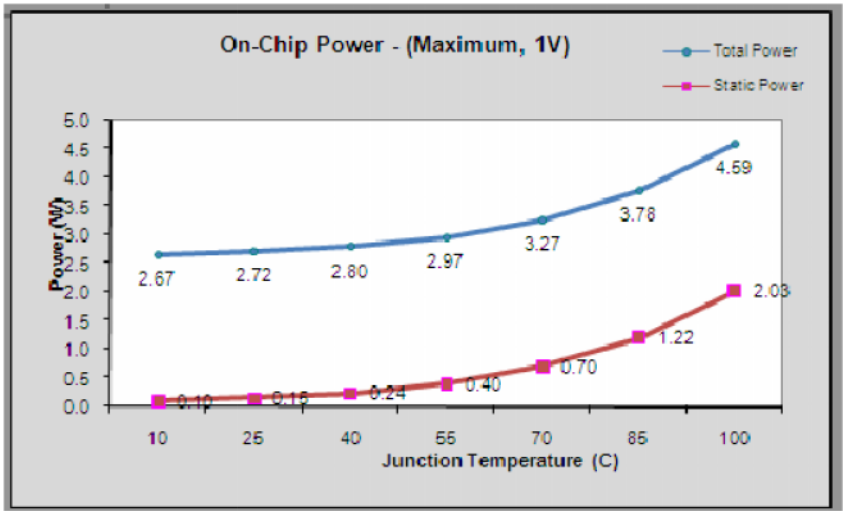

Figure 3.Estimation of Junction temperature VS power for NB-LDPC

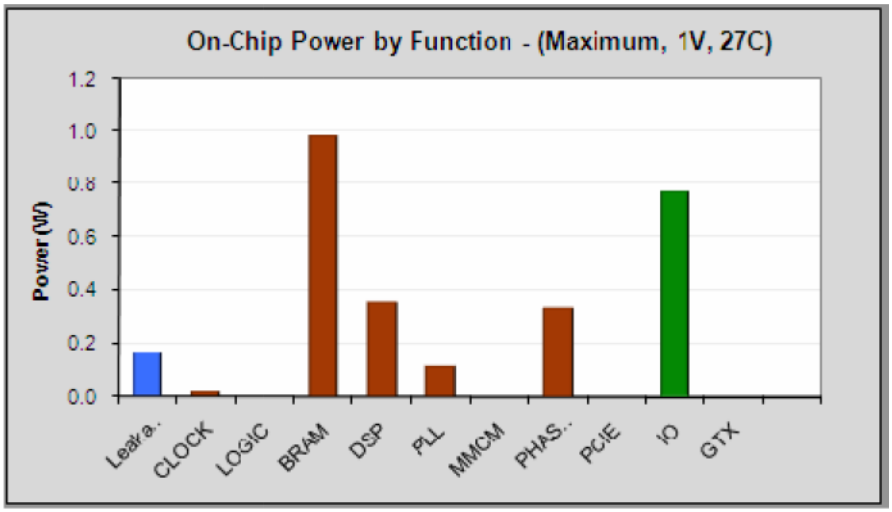

Figure 4.Power consumed by elements in NB-LDPC

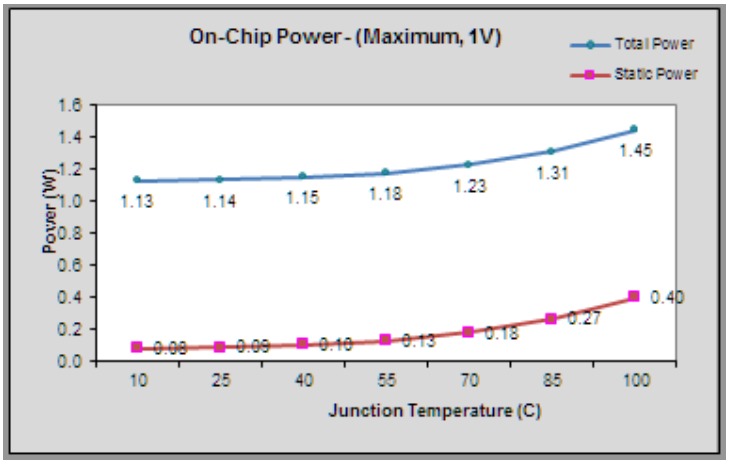

Figure 5.Estimation of Junction Temperature VS Power for EG-LDPC with ML Algorithm

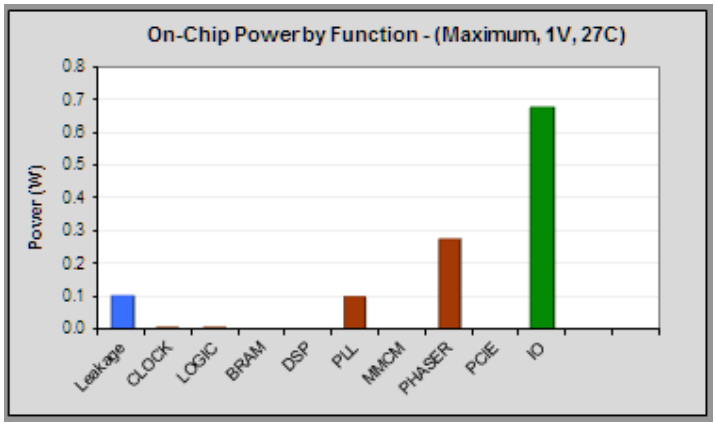

Figure 6.Power Consumed by elements in EG-LDPC with ML Algorithm

5. EXPERIMENTAL RESULT

The Coding for our proposed methodology is written using VHDL coding and simulated using Xilinx 14.2 software. The RTL schematic and Technical schematic of EG-LDPC with ML algorithm is shown in Fig 7 and 8 respectively.

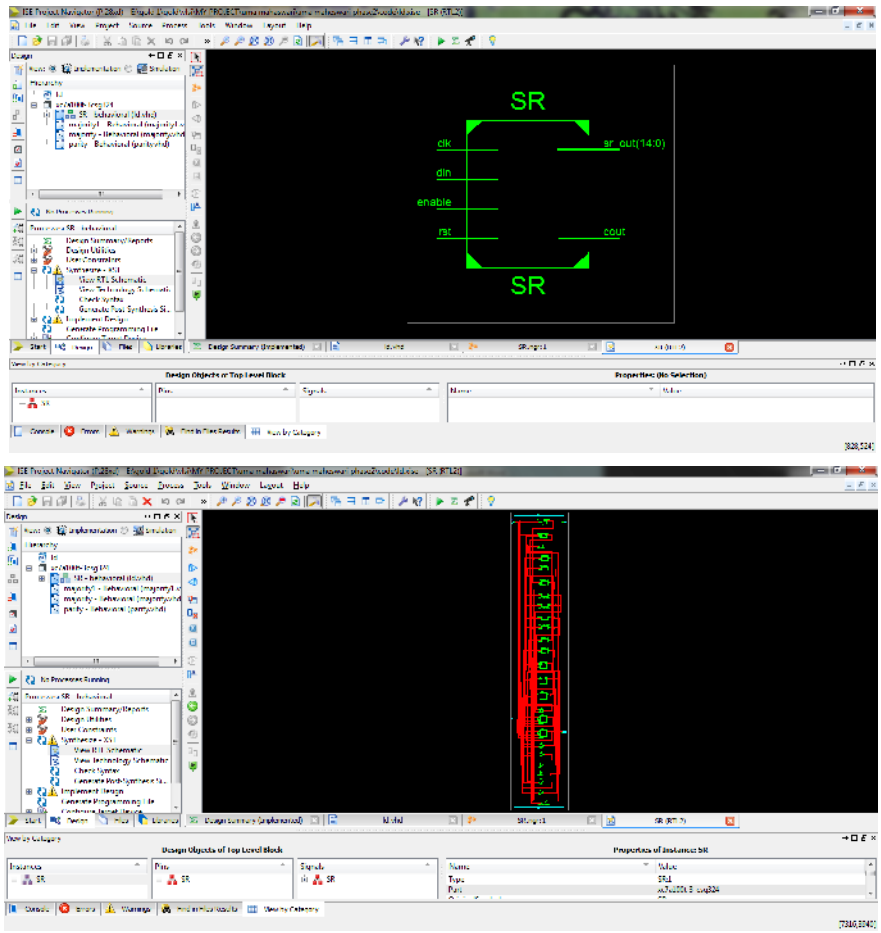

Figure7.RTL Schematic of Proposed work

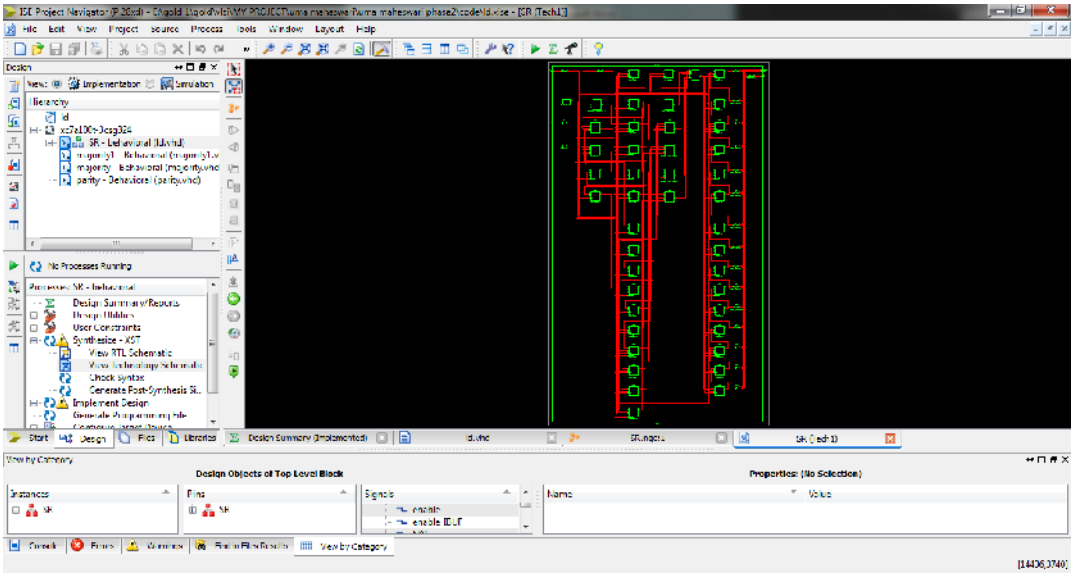

Figure 8.Technical Schematic of Proposed work

6. CONCLUSION

In this Paper, we have proposed ML algorithm for EG-LDPC codes and also compared the performances of both Non-Binary LDPC codes and EG-LDPC using Maximum Likelihood algorithm in terms of power consumption, iterations, junction temperature and thermal margin. The output results shows that EG-LDPC with ML algorithm has better performances than the NB-LDPC codes. The RTL and Technical schematics was also simulated for the proposed system.

REFERENCES

[1] Pedro Reviriego, Juan A. Maestro, and Mark F. Flanagan, “Error Detection in Majority Logic Decoding of Euclidean Geometry Low Density Parity Check (EG-LDPC) Codes,” IEEE Trans. Very Large Scale Integra. (VLSI) syst., vol. 21, no. 1, pp.156-159, Jan. 2013.

[2] Youn Sung Park, Yaoyu Tao, Zhengya Zhang, “A 1.15Gb/s Fully Parallel Nonbinary LDPC Decoder with Fine-Grained Dynamic Clock Gating,” 2013 IEEE International Solid-State Circuits Conference.

[3] S.Naveen, M.Anbuselvi,”FPGA implementation of Non-Binary LDPC Decoderusing Stochastic Computation,” International Journal of Computer Applications (0975 – 8887) National conference on VSLI and Embedded systems 2013.

[4] Laura Conde-canencia, Ali Al Ghouwayel, Emmanuel Boutillon,”Complexity Comparison of Non-Binary Complexity Comparison of Non-Binary LDPC Decoders,” ICT-MobileSummit 2009 Conference Proceedings Poster PaperPaul Cunningham and Miriam Cunningham (Eds)IIMC International Information Management Corporation, 2009ISBN: 978-1-905824-12-0.

[5] Murat Arabaci, Ivan B. Djordjevic, Lei Xu, and Ting Wang, ”Four-Dimensional Nonbinary LDPC-Coded Modulation Schemes for Ultra-High-Speed
Optical Fiber Communication,” IEEE PHOTONICS TECHNOLOGY LETTERS, VOL. 23, NO. 18, SEPTEMBER 15, 2011.

[6] H. Naeimi and A. DeHon, “Fault secure encoder and decoder for memory applications,” in Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst., 2007, pp. 409–417.

[7] S. Liu, P. Reviriego, and J. Maestro, “Efficient majority logic fault detection with difference-set codes for memory applications,” IEEE Trans. Very Large Scale Integra. (VLSI) Syst., vol. 20, no. 1, pp. 148–156, Jan. 2012.

[8] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," presented at the Foundations Nanosci. (FNANO), Snowbird, Utah, 2007.
[9] Sarah J. Johnson, Iterative Error Correction Turbo, Low-Density Parity-Check and Repeat–Accumulate Codes, Cambridge University Press 2010.
[10] Yaoyu Tao, Youn Sung Park, Zhengya Zhang, "High-Throughput Architecture and Implementation ofRegular (2, dc) Nonbinary LDPC Decoders," 2012,IEEE.

Authors

M.Karthick Raja was born in Tuticorin, Tamil Nadu (TN), India, in 1993. He is currently pursuing the Bachelor of Engineering (B.E.)degree with the Department of Electronics and Communication Engineering from the Velammal College of Engineering and Technology, Madurai which was affiliated to Anna University, Chennai, TN, India.

KR.Ragupathy was born in Madurai, Tamil Nadu (TN), India, in 1992. He is currently pursuing the Bachelor of Engineering (B.E.)degree with the Department of Electronics and Communication Engineering from the Velammal College of Engineering and Technology, Madurai which was affiliated to Anna University, Chennai, TN, India.

K.Sathis Kumar was born in Ramnad, Tamil Nadu (TN), India, in 1992. He is currently pursuing the Bachelor of Engineering (B.E.)degree with the Department of Electronics and Communication Engineering from the Velammal College of Engineering and Technology, Madurai which was affiliated to Anna University, Chennai, TN, India.

Mr. M.Sakthivelwas born in Madurai, Tamil Nadu (TN), India, in 1978. He received the Bachelor of Engineering (B.E.) degree with the Department of Electronics and Communication Engineering from the Arulmigu Kalasalingam College of Engineering, TN, India, in 2001 and the Master of Engineering (M.E.) degree in VLSI Design from the Anna University Regional Centre, Madurai, TN, India, in 2013. He worked as Senior Lecturer in KLN Polytechnic College Madurai for 2 and half years, Business Analyst in MIT Kovilpatti for 3 years, Lecturer in Pandian Saraswathi Yadav Engineering College for 4 years. He is currently working as Assistant Professor III in Velammal College of Engineering and Technology from 03-06-2013 to till date. His Teaching Areas includes Object Oriented Programming, Network Security, Computer Hardware and Interfacing, Digital Principles and System Design, Data Structures, Microprocessors and Microcontrollers, Electronics and Microcontroller, Low power VLSI.