# GF(q) LDPC encoder and decoder FPGA implementation using group shuffled belief propagation algorithm

**Fatima Zahrae Zenkouar[1], Mustapha El Alaou[2], Said Najah[1]**
[1]SIA Laboratory, Faculty of Sciences and Technologies, Sidi Mohammed Ben Abdellah University, Fez, Morocco
[2]Laboratory of Computer Science, Signals, Automation and Cognitivism, Department of Physics, Faculty of Sciences Dhar El Mahraz, Sidi Mohamed Ben Abdellah University, Fez, Morocco

**Article Info**

***Article history:***

Received Mar 23, 2021
Revised Nov 6, 2021
Accepted Dec 4, 2021

***Keywords:***

AWGN channel
FPGA
GSBP algorithm
NB LDPC code

**ABSTRACT**

This paper presents field programmable gate array (FPGA) exercises of the GF(q) low-density parity-check (LDPC) encoder and interpreter utilizing the group shuffled belief propagation (GSBP) algorithm are presented in this study. For small blocks, non-dual LDPC codes have been shown to have a greater error correction rate than dual codes. The reduction behavior of non-binary LDPC codes over GF (16) (also known as GF(q)-LDPC codes) over the additive white Gaussian noise (AWGN) channel has been demonstrated to be close to the Shannon limit and employs a short block length (N=600 bits). At the same time, it also provides a non-binary LDPC (NB-LDPC) code set program. Furthermore, the simplified bubble check treasure event count is implemented through the use of first in first out (FIFO), which is based on an elegant design. The structure of the interpreter and the creation of the residential area he built were planned in very high speed integrated circuit (VHSIC) hardware description language (VHDL) and simulated in MODELSIM 6.5. The combined output of the Cyclone II FPGA is combined with the simulation output.

***Corresponding Author:***

Fatima Zahrae Zenkouar
SIA Laboratory, Faculty of Sciences and Technologies, Sidi Mohammed Ben Abdellah University
B.P. 2202-Route Imouzzer, Fez, Morocco
Email: fatimazahrae.zenkouar@usmba.ac.ma

## 1. INTRODUCTION

The power of low-density parity-check (LDPC) codes based on the finite high-order field GF(q) has long been recognized. For short and medium codeword lengths, these codes increase binary LDPC performance. Non-binary LDPC (NB-LDPC) codes have been demonstrated to outperform turbo convolutional codes (TCC) and binary LDPC codes, retaining the advantages of steep drop zone and low error of short codewords (typical TCC) (typical binary LDPC) [1]–[3]. This gain, however, comes at the expense of greater decoding difficulty. Indeed, as q rises, the complexity of the decoder rises, limiting design options and encouraging the search for simpler decoding algorithms.

In recent years, several efforts have been made to reducing the convolution of NB-LDPC decoders, and divers associated architecture algorithms have been proposed. [1], [4]–[6] have proposed an LDPC code decoding algorithm with reduced complexity. This algorithm is known as the group shuffled belief propagation (GSBP) extended minimum sum algorithm, and it is based on the minimum sum (MS) algorithm's generalization. To decrease the computational complexity of updating the control node, the approach involves using a restricted number of nm-reliabilities in the message at the control node's input [7]–[9].

A new GSBP decoder implementation has been suggested. This technique is unique in that it solves the memory problem of non-binary LDPC decoders while drastically decreasing the complexity per iteration. The new GSBP decoder's major feature is that it extends the truncation concept of vector messages to control and data node inputs. To decrease the impact of messages on code speed, the authors effectively shortened messages from $q$ to $nm$. They also have a good offset adjustment to make up for the performance loss.The GSBP decoder's complexity is now theoretically dominated by $O\ (nm.\log nm)$, with $nm << q$, which is a significant decrease in complexity over all previous techniques [10]–[15].

Our research is based on the GSBP algorithm [10], [16]–[20] This decreases the number of GF elements that must be processed in each decoding phase from $q$ to $nm$, nm and $lt$; $q$, while providing excellent functional performance. We convert the GSBP algorithm into a low latency, prefetching elementary NC enquiry control number (ECN) that relaxes redundancy control at low error rates while sacrificing only a tiny amount of functional performance. A new method is presented to bring variable node (VN) latency closer to that of check node (CN), in fact this design is based on the significant reduction of combinatorial optimization search and counting time. To achieve the highest possible pipeline efficiency, we suggest a conflict-free memory to handle the data dependencies caused by unstructured codes. A full (2, 4) regular, (960, 80) GF (6) LDPC decoder is prototyped on a Cyclone IV EP CE115F29C8 field programmable gate array (FPGA) embedded on the Altera DE2115 board. The decoder achieves good error correction performance.

We presented this article at the following section 2 presents the symbol model, LDPC codes over GF(q). Section 3 presents the implementation LDPC: design architecture of GSBP. Section 4 presents the FPGA results and prototype, and we conclude in section 5.

## 2. SYMBOL MODEL, LDPC CODES OVER GF(q)

Linear block codes, often known as LDPC codes, are a kind of linear block coding. Non-binary LDPC codes over $GF(q)$ are considered binary LDPC codes over $GF(q)\ if\ q = 2\ (2)$. A vector space projected on $GF(q)$ is used to define the elements in $GF(q)$. Choose $q$ as:

$$q = 2^m \tag{1}$$

The code is given and its value *r* is supplied as fallows using the ultra-sparse parity check matrix H, where $m$ is a positive integer and $m > 1$.

$$R = (N - M)\ /\ N \tag{2}$$

$M = N - K$, where N denotes codeword length and K denotes message length. A column weight of at least two and a row weight that is as uniform as feasible are used to construct the sparse parity check matrix H of size (M × N). This building method ensures:

- Every column has a certain number $\gamma$ of items
- Every line has a number of elements ρ
- Non-zero elements occur in either row or column at more than one position in any two rows or columns.

According to the first two, H has a constant row and column weight and forms typical LDPC code. The third asserts that the entire graph of code is devoid of a four-cycle cycle. First permutation matrices are produced utilizing non binary components in $GF(q)$ and dispersed in the base matrix to build parity check matrix H in the creation of LDPC codes. In the sparse parity check matrix H, an LDPC is a linear block code with a low density of none zero entries. Tanner graphs, also known as bipartite graphs, are used to depict these codes, as shown in Figure 1.

### 2.1. System model

Figure 2 shows an NB-LDPC coded modulation system. The LDPC encoder is used to encode the k information bits in the input message vector $u\ GF(q)$ into a codeword, resulting in an encoded message vector $v$ of length N coded bits. The equation gives the rate of the LDPC encoder (2). These bits are modulated into QPSK symbols (represented by the letter X) and then delivered across the additive white Gaussian noise (AWGN) channel. The received signal *r* is written as:

$$r = X + n \tag{3}$$

where $n$ denotes AWGN noise and is modulated by a zero mean Gaussian. The following is a random sequence with variance:

$$\sigma^2 = (2R\, E_b/N0)^{-1} \quad (4)$$

where Eb/N0 is SNR.

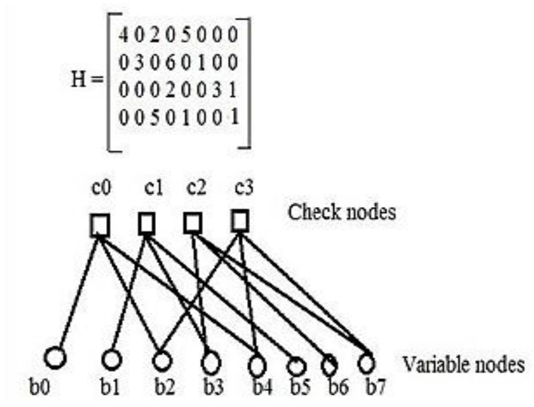


Figure 1. The tanner graph over GF and the parity check matrix H (8)

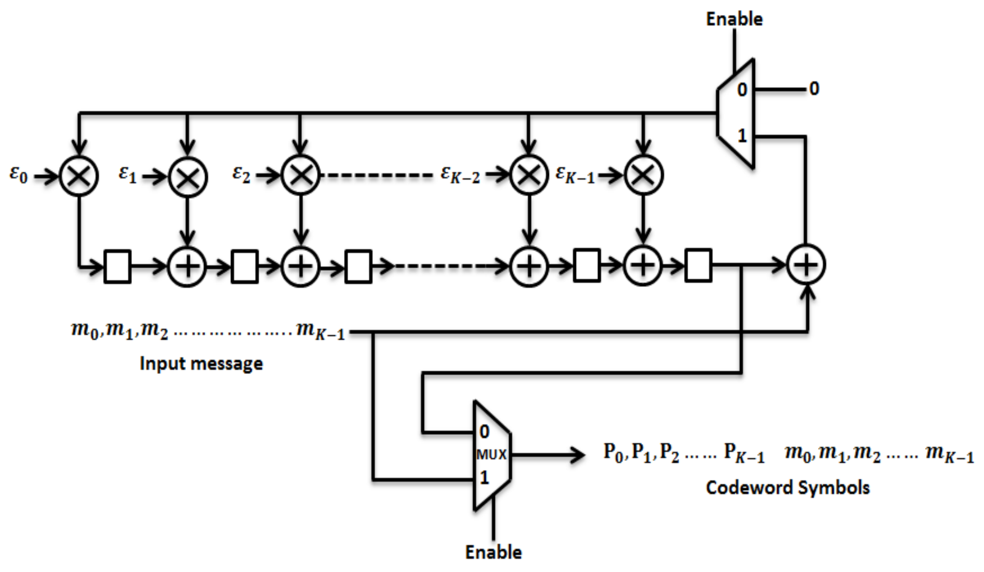


Figure 2. Encoder global block

## 3. IMPLEMENTATION LDPC: DESIGN ARCHITECTURE OF GSBP

### 3.1. Encoding

An M × N parity check matrix H defines non-binary LDPC codes, components that are defined over a finite field GF(q). The parity check matrix can take on q − 1 values for each non-zero member, resulting in a parity check matrix. Because the H is not in a logical order at first, we write it as:

$$H = [P|Im] \quad (5)$$

where $Im$ denotes the M×M identity matrix and P denotes the M×K dimension matrix, with $K = N - M$. The GF is used to do all arithmetic operations (q). With dimensions K x N, a generator matrix can be created:

$$G = [Ik|P'] \quad (6)$$

The encoder turns a message frame U from GF(q) containing K information bits into a codeword V of length N (N symbols).

$$V = G.U \tag{7}$$

The matrix multiplication is done over the finite field GF in this case (q). Finally, the encoder block sends us the codeword V, which has a length of $N = nP\ bits$. For the multiplication we use the efficient digit serial Karatsuba multiplication method and it uses (3dm/2) AND gate,6 m+n+ (3dm/2)+m2 +d-7 XORs and 3 m-3 registers as illustrated in Figure 3.

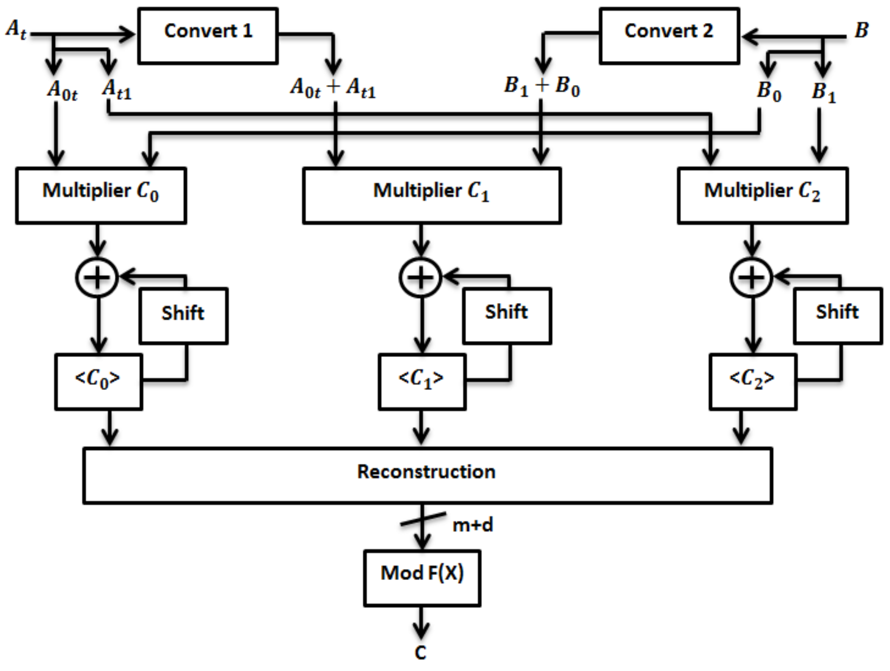


Figure 3. Efficient digit-serial Karatsuba multiplication

### 3.2. Top level architecture of decoder

When it comes to decoding non-binary codes, there are two issues to consider: The first issue is the iterative processing of control and variable nodes, and the second issue is identifying the most likely codeword $v$ that meets the criteria $v\ H = 0$, with a probability of v determined using the channel model. The Figure 4 shows the top-level architecture design.

The GSBP algorithm is an extension of the MinSum algorithm from binary codes to NBLDPC codes. Vectors of likelihood ratio values are transferred between the VN and CN processors in the messages log-likelihood ratios (LLRs). The extended min sum GSBP algorithm has recently been proposed for non-binary LDPC decoding. This novel technique can reduce the number of comparison operations by a factor of 3, resulting in a lower hardware complexity, without introducing any significant performance degradation. A message in the GSBP algorithm is a vector of $q$ sub-messages. Let $xj$ be the code symbol for the code word's j-th character.

Let $\lambda j = [\lambda j(0), \lambda j(1), \ldots, \lambda j\ (q-1)]$ be the *a priori* channel information for $xj$. The sub-message $\lambda j$ is a log-likelihood ratio (LLR) defined as $\lambda j\ (d) = log\ (Prob\ (xj = zj)/Prob\ (xj = d))$, where $zj$ is the most likely (ML) symbol for $xj$. We denote $\alpha i,j$ and $\beta i,j$ as the V2C and C2V soft messages passed between the $i$-th CN and $j$-th VN respectively.

Let $xi, j = hi, j \otimes xi$.

For the $i$-th CN, let the configuration Li $(xi, j = d)$ be the sequence such that $xi$, $j$=$d$ and the $i$-th check-sum is satisfied. Define an $s$-truncated configuration such for each $j \in Ni\backslash j, \alpha i, j\ (xi, j)$ is of the $s$ smallest sub-messages of $\alpha i, j(d)$ over all $\in GF(q)$. Let $Li(xi, j = d|s)$ be the set of the $s$-truncated configurations. Taking $s < q$ necessitates extra procedures known as message truncation, which involves sorting the sub-messages and ignoring the $q - s$ biggest ones. Let $\kappa$ and $\kappa$max denote the iteration counter and the maximum number of iterations respectively.

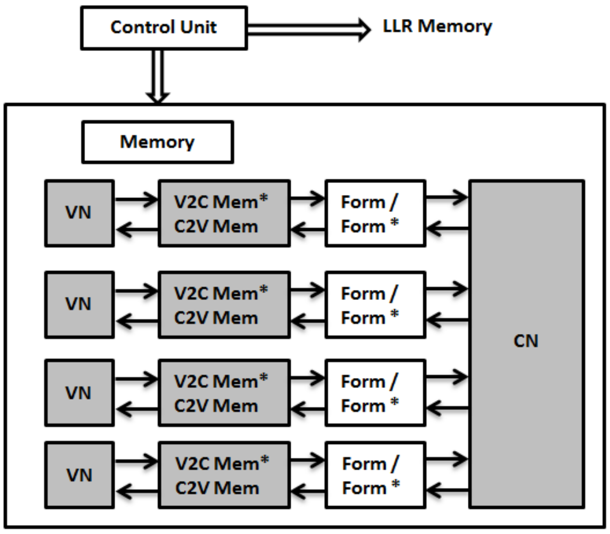


Figure 4. Top-level architecture design

## 3.3. Definition of NB LLR values

The first step in the Min-Sum technique is to calculate the LLR value for each symbol in the codeword. With the premise that the GF(q) symbols are equiprobable [21] yields the LLR value Lk (x) of the $k$-th symbol:

$$Lk(x) = ln\ [P\ (yk|\tilde{}xk)/\ P\ (yk|x)] \qquad (8)$$

where $\tilde{}xk$ is the symbol of GF(q) that maximizes $P\ (yk\ |x), i.e. \tilde{}xk = arg\ maxx \in GF(q)\ , \{P\ (yk\ |x)\}$ and $yk$ is the received symbol. Note that $Lk\ (\tilde{}xk) = 0$ and, for all x ∈ $GF\ (q), Lk\ (x)\ \geq\ 0$. When a result, as a symbol's LLR grows, its dependability diminishes. When addressing the finite precision representation of the LLR values, this LLR formulation eliminates the requirement to re-normalize the messages after each node update calculation and reduces the effect of quantization. The Figure 5 shows a block diagram of the LLR computation and the Figure 6 shows a timing diagram of the LLR computation over $GF\ (16). Nm = 10$.

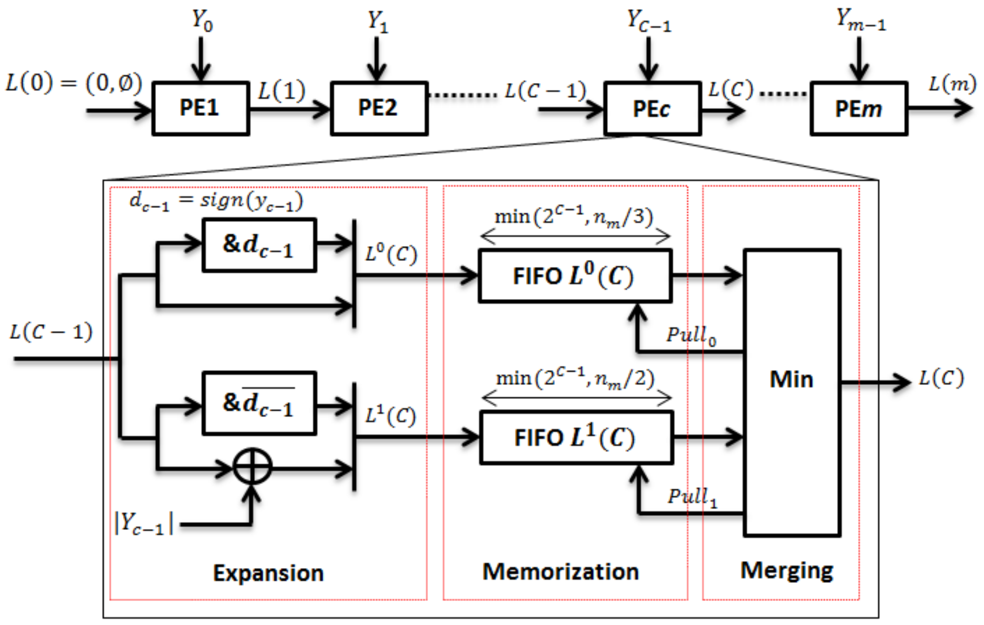


Figure 5. Block diagram of the LLR computation

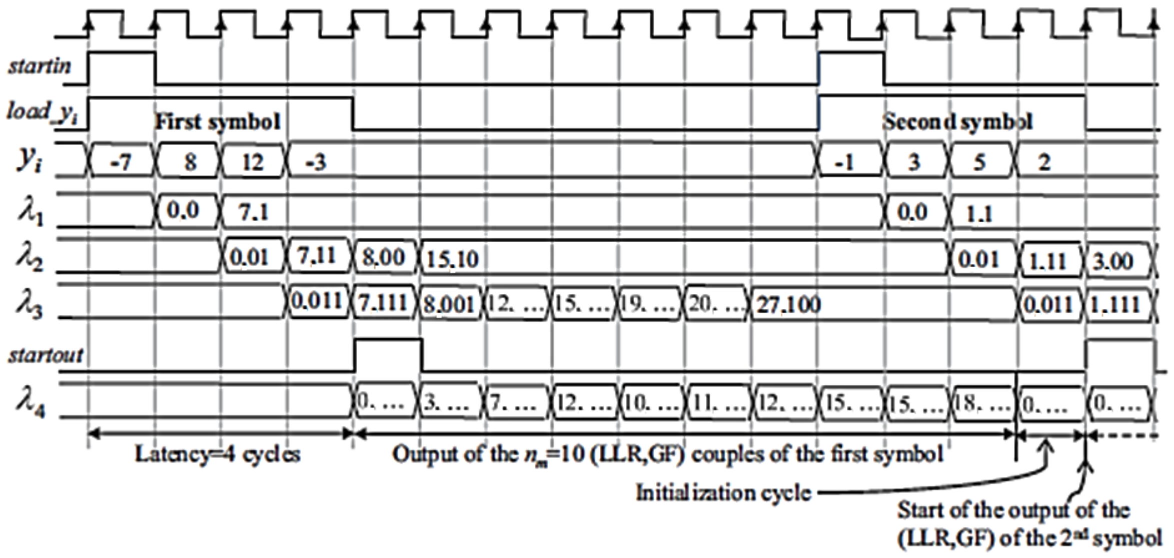


Figure 6. Timing diagram of the LLR computation over $GF(16)\ Nm = 10$

## 3.4. Check node architecture

The check node processor (CNP) may be constructed using either the FB architecture or a tree-based structure [22]. The tree structure has the advantage of reducing the number of ECN in the critical route to a bare minimum and ensuring consistency across all outputs. We considered the Tree structure in our study for these reasons.

The symbols of the messages entering the CNP must be multiplied by the non-zero members of the parity check matrix as illustrated in Figure 7. In addition to the CNP's output messages that are split by these non-zero elements (the row corresponding to the CNP in the Tanner graph). As a result, the implemented CNP architecture performs multiplications on GF(q) using the hardwired multipliers described in [23]–[25].

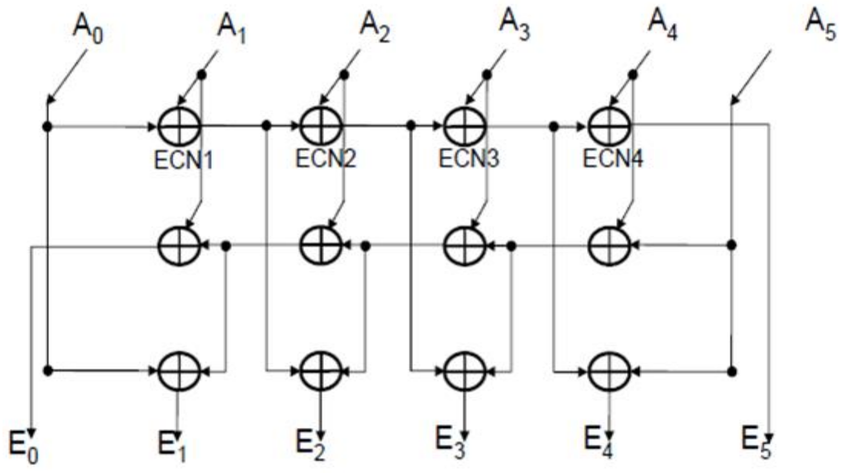


Figure 7. Design of the CN block

## 3.5. Variable node architecture

We start the VN operations immediately after the CN operation, a transfer of messages from C to V in the memory. The message C to V is computed from the messages C to V and the previous message V to C taking into account the GF indices that must match. To reduce latency, we allocate only $L(svn)$, the length of the sorter used in VN, to scan the vector L. Due to a lack of space, we just name the sorter in the memory. Due to space constraints, we will only provide the algorithm and leave the rest of the discussion for a subsequent publication. The Figure 8 shows architecture of the VN block and the Figure 9 shows architecture of the sorter block in VN.

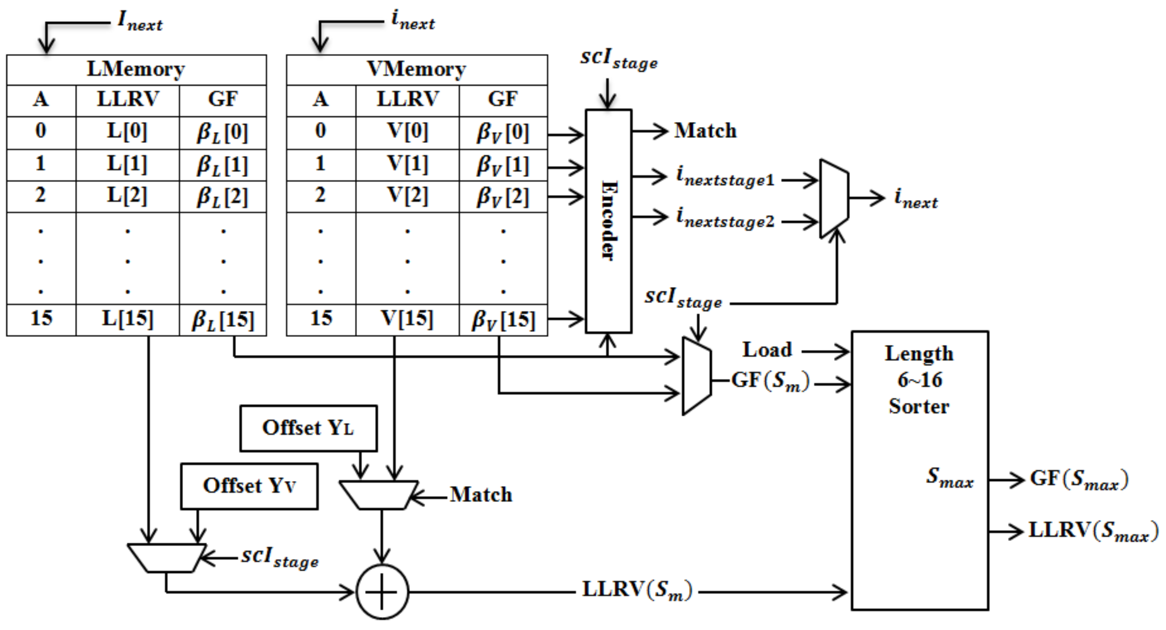


Figure 8. Architecture of the VN block

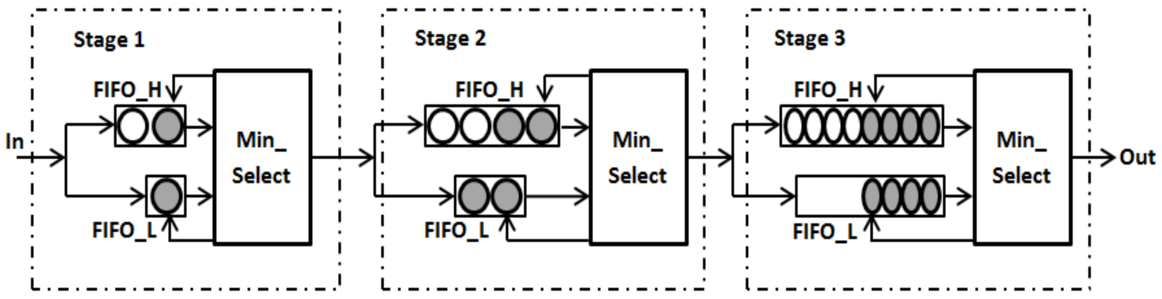


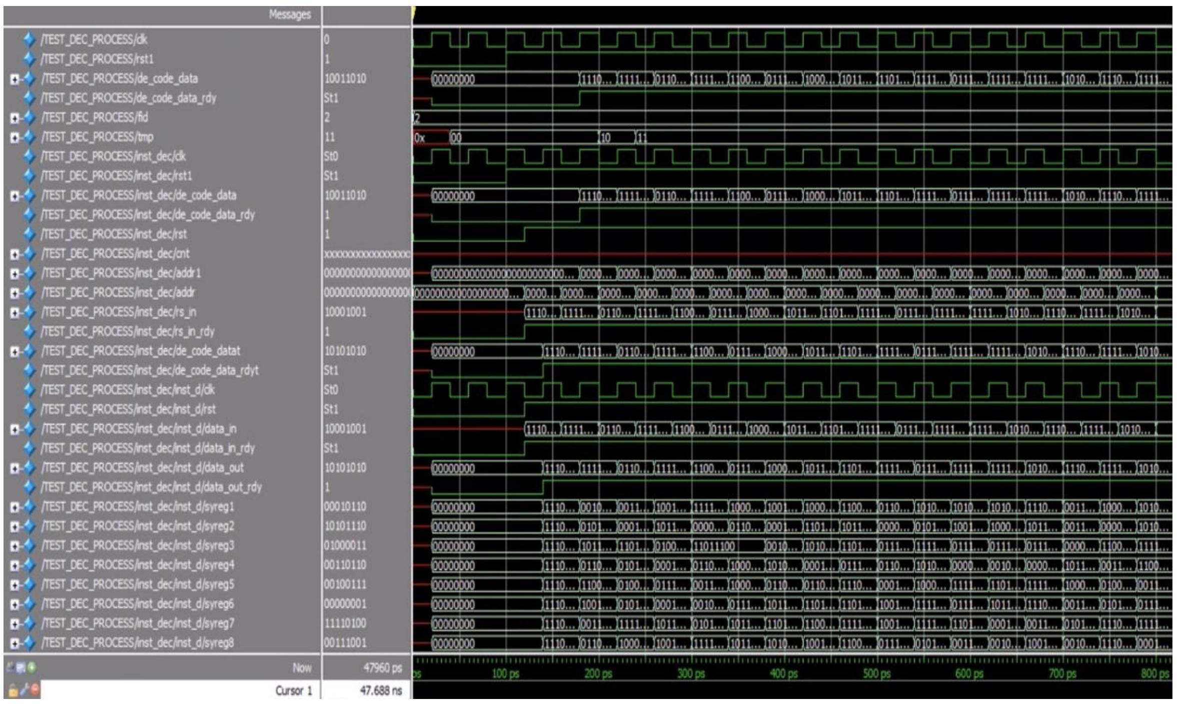
Figure 9. Architecture of the sorter block in VN

## 4. FPGA RESULTS AND PROTOTYPE

On a Cyclone IV FPGA, the suggested decoder architecture was prototyped for a (2, 4)-regular (960, 480) NB-LDPC code over GF (64). The FPGA decoder has 10 decoding cycles and runs at 100 MHz with a code rate of 2.44 Mbps. Based on the suggested design of this FPGA device, we were able to map a semi complete allied decoder. This result is a substantial improvement over the current GSBP decoder implementation. The layer decoding improves convergence and replicate show that the average number of iterations for FER=105 decreases from 2.39 to 1.77. The Table 1 gives an overview of how they look time required for input and output. The Figure 10 illustrate the decoding output and the Figure 11 illustrate the performances of NB-LDPC over GF (64) using the GSBP algorithm.

Table 1. Summary of the timing

| Delay | 8.487 ns |
|---|---|
| Maximum frequency | 117.830 MHz |
| Minimum input arrival time before clock | 5.093 ns |
| After the clock, the maximum output necessary | 12.119ns |



```
Messages
/TEST_DEC_PROCESS/clk                              0
/TEST_DEC_PROCESS/rst1                             1
/TEST_DEC_PROCESS/de_code_data                     10011010
/TEST_DEC_PROCESS/de_code_data_rdy                 St1
/TEST_DEC_PROCESS/fid                              2
/TEST_DEC_PROCESS/tmp                              11
/TEST_DEC_PROCESS/inst_dec/clk                     St0
/TEST_DEC_PROCESS/inst_dec/rst1                    St1
/TEST_DEC_PROCESS/inst_dec/de_code_data            10011010
/TEST_DEC_PROCESS/inst_dec/de_code_data_rdy        1
/TEST_DEC_PROCESS/inst_dec/rst                     1
/TEST_DEC_PROCESS/inst_dec/cnt                     xxxxxxxxxxxxxxxxx
/TEST_DEC_PROCESS/inst_dec/addr1                   00000000000000000
/TEST_DEC_PROCESS/inst_dec/addr                    00000000000000000
/TEST_DEC_PROCESS/inst_dec/rs_in                   10001001
/TEST_DEC_PROCESS/inst_dec/rs_in_rdy               1
/TEST_DEC_PROCESS/inst_dec/de_code_datat           10101010
/TEST_DEC_PROCESS/inst_dec/de_code_data_rdyt       St1
/TEST_DEC_PROCESS/inst_dec/inst_d/clk              St0
/TEST_DEC_PROCESS/inst_dec/inst_d/rst              St1
/TEST_DEC_PROCESS/inst_dec/inst_d/data_in          10001001
/TEST_DEC_PROCESS/inst_dec/inst_d/data_in_rdy      St1
/TEST_DEC_PROCESS/inst_dec/inst_d/data_out         10101010
/TEST_DEC_PROCESS/inst_dec/inst_d/data_out_rdy     1
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg1           00010110
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg2           10101110
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg3           01000011
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg4           00110110
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg5           00100111
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg6           00000001
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg7           11110100
/TEST_DEC_PROCESS/inst_dec/inst_d/syreg8           00111001
Now                                                47960 ps
Cursor 1                                           47.688 ns
```

Figure 10. Decoding output

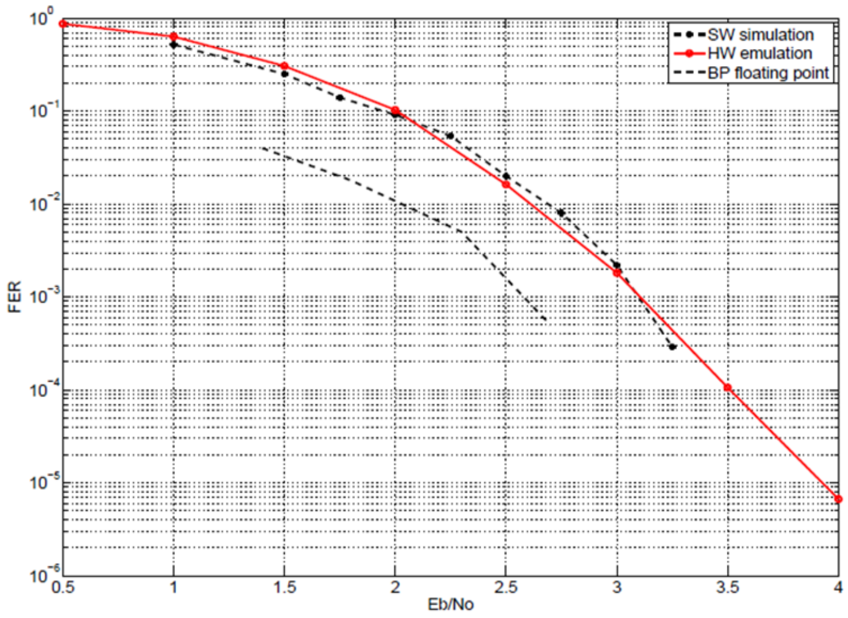


Figure 11. Performances of NB-LDPC over GF (64) using the GSBP algorithm

## 5. CONCLUSION

A hardware simplified decoding algorithm using the simplified extended min sum algorithm was favorably presented. The concept was centered on considerably decreasing the combinatorial optimization search space as well as the computing time. To minimize latency and enable effective pipeline scheduling, VN and CN designs based on skimming, prefetching, and easing redundancy control are proposed. To minimize pipeline delays, a conflict-free memory was utilized to handle data risks. The results suggest that the decoding is working well. Compared to the GSBP method for NB-LDPC codes, the design has considerably reduced computational complexity and memory use, according to our research.

## REFERENCES

[1] R. G. Gallager, *Low-density parity-check codes*. The MIT Press, 1963.

[2] W. Sulek, M. Kucharczyk, and G. Dziwoki, “GF(q) LDPC decoder design for FPGA implementation,” in *2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013*, Jan. 2013, pp. 460–465, doi: 10.1109/CCNC.2013.6488484.

[3] C. Spagnol, E. M. Popovici, and W. P. Marnane, “Hardware implementation of GF(2m) LDPC decoders,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 12, pp. 2609–2620, Dec. 2009, doi: 10.1109/TCSI.2009.2016621.

[4] M. M. Mansour and N. R. Shanbhag, “Memory-efficient turbo decoder architectures for LDPC codes,” in *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation*, 2002, pp. 159–164, doi: 10.1109/SIPS.2002.1049702.

[5] M. M. Mansour and N. R. Shanbhag, “Low-power VLSI decoder architectures for LDPC codes,” in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2002, pp. 284–289, doi: 10.1109/LPE.2002.146756.

[6] T. Zhang and K. K. Parhi, “A 54 Mbps (3,6)-regular FPGA LDPC decoder,” in *IEEE Workshop on Signal Processing Systems*, 2002, pp. 127–132, doi: 10.1109/SIPS.2002.1049697.

[7] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, “VLSI architectures for iterative decoders in magnetic recording channels,” *IEEE Transactions on Magnetics*, vol. 37, no. 2, pp. 748–755, Mar. 2001, doi: 10.1109/20.917611.

[8] Y. Kou, S. Lin, and M. P. C. Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711–2736, 2001, doi: 10.1109/18.959255.

[9] S. H. Kang and I. C. Park, “Loosely coupled memory-based edcoding architecture for low density parity check codes,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 5, pp. 1045–1056, May 2006, doi: 10.1109/TCSI.2005.862181.

[10] M. M. Mansour and N. R. Shanbhag, “On the architecture aware structure of LDPC codes from generalized Ramanujan graphs and their decoder architecture,” in *Proc. 37th Annu. Conf. Inf. Sci. Syst.*, 2003, pp. 215–220.

[11] D. E. Hocevar, “LDPC code construction with flexible hardware implementation,” in *IEEE International Conference on Communications*, 2003, vol. 4, pp. 2708–2712, doi: 10.1109/icc.2003.1204466.

[12] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974, doi: 10.1109/TIT.1974.1055186.

[13] Y. Kou, S. Lin, and M. P. C. Fossorier, “Low density parity check codes based on finite geometries: a rediscovery,” in *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, p. 200, doi: 10.1109/ISIT.2000.866498.

[14] E. Yeo, P. Pakzad, B. Nikolić, and V. Anantharam, “High throughput low-density parity-check decoder architectures,” in *Conference Record/IEEE Global Telecommunications Conference*, 2001, vol. 5, pp. 3019–3024, doi: 10.1109/glocom.2001.965981.

[15] S. T. Brink, G. Kramer, and A. Ashikhmin, “Design of low-density parity-check codes for modulation and detection,” *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, Apr. 2004, doi: 10.1109/TCOMM.2004.826370.

[16] S. Y. Chung, T. J. Richardson, and R. L. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 657–670, 2001, doi: 10.1109/18.910580.

[17] T. J. Richardson and R. L. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001, doi: 10.1109/18.910577.

[18] M. M. Mansour and N. R. Shanbhag, “Turbo decoder architectures for low-density parity-check codes,” in *Global Telecommunications Conference*, 2002, vol. 2, pp. 1383–1388, doi: 10.1109/GLOCOM.2002.1188425.

[19] E. Boutillon, L. Conde-Canencia, and A. Al Ghouwayel, “Design of a GF(64)-LDPC decoder based on the EMS algorithm,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 10, pp. 2644–2656, Oct. 2013, doi: 10.1109/TCSI.2013.2279186.

[20] M. C. Davey and D. MacKay, “Low-density parity check codes over GF(q),” *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, Jun. 1998, doi: 10.1109/4234.681360.

[21] D. Sridhara, R. M. Tanner, and T. E. Fuja, “Low density parity check codes from permutation matrices,” in *The 35th Annual Conference on Information Sciences and Systems (CISS)*, 2001.

[22] M. G. Luby, M. Amin Shokrolloahi, M. Mizenmacher, and D. A. Spielman, “Improved low-density parity-check codes using irregular graphs and belief propagation,” 1998, doi: 10.1109/ISIT.1998.708706.

[23] A. Bennatan and D. Burshtein, “Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels,” *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 549–583, Feb. 2006, doi: 10.1109/TIT.2005.862080.

[24] H. Wymeersch, H. Steendam, and M. Moeneclaey, “Log-domain decoding of LDPC codes over GF(q),” in *IEEE International Conference on Communications*, 2004, vol. 2, pp. 772–776, doi: 10.1109/icc.2004.1312606.

[25] C. Poulliat, M. Fossorier, and D. Declercq, “Design of regular (2, $d_c$)-LDPC codes over GF(q) using their binary images,” *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626–1635, Oct. 2008, doi: 10.1109/TCOMM.2008.060527.

## BIOGRAPHIES OF AUTHORS

**Fatima Zahrae Zenkouar** she received a state engineering degree in embedded systems and industrial data from the National School of Applied Sciences, University of Sidi Mohammed Ben Abdellah, Fez, Morocco in 2015. He is currently pursuing his Ph.D. degree in Computer Science with the Laboratory of Intelligent Systems and Application at the Faculty of Science and Technology of Fez. His research interests include signal/data processing, code theory, parallel computing, and LDPC algorithms. She can be contacted at email: fzenkouar@gmail.com.

**Mustapha El Alaoui** is born in the Old Medina, Fes, Morocco, 1994. He received his Master degree since 2017 in Micro-Electronics in Faculty of Sciences Dhar EL Mahraz (FSDM), Sidi Mohammed Ben Abdellah University (USMBA), Fez, Morocco. He received a Ph.D degree in Electrical Engineering in 2021 from Laboratory of Computer Science, Signals, Automation and Cognitivism (LISAC), Department of Physics, FSDM, USMBA, Fez, Morocco. His research interests include Li-Ion battery charger interface (BCI) and BMS, RFID passive and active tags, CMOS mixed mode integrated circuit design, integrated class-D power output stage and renewable energy. He can be contacted at email: mustapha.elalaoui@usmba.ac.ma.

**Said Najah** He received a Ph.D. degree in Computer Science from the Faculty of Science, University Sidi Mohamed Ben Abdellah, Fez, Morocco in 2006. He is currently a professor of the Department of Computer Science, Faculty of Science and Technology Fez Morocco. He is a member in the laboratory of intelligent systems and application (LSIA). His current research interests include parallel computing, code theory, signal processing and artificial intelligence. He can be contacted at email: said.najah@usmba.ac.ma.