Hardware Implementation Analysis of Min-Sum Decoders

Rajagopal ANANTHARAMAN[1], *Karibasappa KWADIKI*[2], *Vasundhara Patel KEREHALLI SHANKAR RAO*[3]

[1]Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Visvesvaraya Technological University, VTU Main Road, Belgaum, Karnataka 590018, India
[2]Department of Electronics and Communication Engineering, Dayananda Sagar Academy of Technology and Management, Visvesvaraya Technological University, VTU Main Rd, Belgaum, Karnataka 590018, India
[3]Department of Electronics and Communication Engineering, BMS College of Engineering, Visvesvaraya Technological University, VTU Main Road, Belgaum, Karnataka 590018, India

gopiluck@gmail.com, karibasappably@gmail.com, vasu.ece@bmsce.ac.in

DOI: 10.15598/aeee.v17i2.3042

Abstract. *The objective of this work is to propose a modified Min-Sum decoding Low Density Parity Check (LDPC) algorithm and perform the hardware implementation analysis of Min-Sum, optimized Min-Sum and modified Min-Sum decoders. The Min-Sum algorithm mainly uses the process of finding the minimum and addition. Hence the number of multiplications is drastically reduced which helps in reducing the complexity of implementation. Adding an optimisation factor to the decoder increases the accuracy and reduces the number of iterations required to compute the decoded message. Hence the process of optimisation reduces the overall decoding time required. Modified Min-Sum algorithm is proposed to further improve the performance by decreasing the number of stages in the decoding process which reduces the complexity in Field Programmable Gate Array (FPGA) implementation.*

Keywords

Bit Error Rate (BER), Field Programmable Gate Array (FPGA), Logarithmic Sum Product (LogSP), Low Density Parity Check (LDPC), Signal to Noise Ratio (SNR), Sum Product Algorithm (SPA).

1. Introduction

Low-Density Parity Check (LDPC) codes are a type of error-correction codes first introduced by Gallager way back in 1962 [1] and [2] during his research work. However, due to the lack in availability of advanced computing methods, practical implementation was not possible. With the advancement in computing capabilities and the advent of new theories finally, Mackay and Neal re-invented LDPC codes in 1996 [3], [4] and [5]. LDPC codes are a type of linear block codes made up of sparse parity-check matrices. LDPC codes show better performance, greater flexibility and parallel capability which ensures better hardware implementation [6]. Because of the wonderful error-correction capabilities, LDPC codes have become the most accepted technique in all modern applications in the communication field [7].

The basic decoding algorithm proposed by Gallager in 1962 was Sum Product Algorithm (SPA). SPA [8] and [9] comprises of more addition and multiplication steps which makes the computation complex in case of real-time implementations such as Field Programmable Gate Array (FPGA) [10], [11] and [23]. As a result, a simpler version of the SPA was introduced to facilitate practical implementation of codes. The Min-Sum is one such algorithm designed to reduce hardware complexity [12]. In this method, approximation is done at check nodes for complex computation by comparing and summation operation. N. Weiberg [12] worked extensively on Min-Sum algorithm and showed that it greatly reduces the implementation complexity. Several modified versions of Min-Sum decoding algorithms were later proposed to improve its performance with respect to SPA [13], [14], [15], [16], [17], [21] and [22]. Min-Sum decoding technique is greatly used in many digital broadcasting applications due to its low complexity. Here, in case of optimized Min-Sum [15], a scaling factor is used to decrease the error caused by

having minimum operation and thereby increasing the accuracy. A modified Min-Sum algorithm for LDPC is proposed in this work. Its performance is very close to SPA while retaining its main feature, i.e. low complexity. The Min-Sum algorithm performs in main four steps [12] and [13]:

- Initialization.
- Horizontal Step.
- Vertical Step.
- Decoding / Estimation.

1.1. Initialization

In initialization step, the value of $\vec{L}(c_i)$ is obtained from the received vector $\vec{r}$. This received data has noise added to it. The received data is negated and assigned to $\vec{L}(c_i)$. A priori information is:

$$\vec{L}(c_i) = \log\left(\frac{P_r(C_i = 0 \mid y_i)}{P_r(C_i = 1 \mid y_i)}\right) \qquad (1)$$
$$\vec{L}(c_i) = 2R_n/\sigma^2.$$

In the BI-AWGN Channel, the above value can be replaced by the following equation.

$$\vec{L}(c_i) = -R_n. \qquad (2)$$

The advantage of this process is that the noise power σ^2 is not required for computation.

After obtaining $\vec{L}(c_i)$ bit to check message, initialization is done. In this process, the value of $\mathbf{L}(q_{ij})$ is obtained which is the initialized matrix of dimension equal to parity check matrix. To obtain $\mathbf{L}(q_{ij})$ matrix, the vector $\vec{L}(c_i)$ is multiplied with each row of parity check matrix $\mathbf{H}$.

$$\mathbf{L}(q_{ij}) = \log\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right), \qquad (3)$$

$$\mathbf{L}(q_{ij}) = \mathbf{H}. * \vec{L}(c_i). \qquad (4)$$

For error correction and detection at the receiver, the same $\mathbf{H}$ matrix, which is used at the Transmitter side, is used.

Then $\mathbf{L}(q_{ij})$ is factorized into its sign and magnitude. For that, the sign and magnitude values of the previous step are extracted and assigned to two variables of different matrices [8].

$$\mathbf{L}(q_{ij}) = \alpha_{ij} \cdot \beta_{ij}, \qquad (5)$$

Where:

$$\alpha_{ij} = \text{sign}\left[\mathbf{L}(q_{ij})\right], \qquad (6)$$
$$\beta_{ij} = \mid \mathbf{L}(q_{ij}) \mid . \qquad (7)$$

1.2. Horizontal Step

In this step, the check node is updated by finding the minimum of all the variable nodes, multiplying it by the sign and assigning it to the check node. Also, in this step, a message from the variable node to the check node is passed. Thus, variable node data is passed on to the check node. From α_{ij} and β_{ij} matrices, $\mathbf{L}r_{ij}$ matrix is obtained by updating the check node. To update the check node, the minimum value (except the one being updated) is calculated in each row of β_{ij} matrix and multiplied it with the product of signs of the same row from α_{ij} matrix. This value is stored in $\mathbf{L}r_{ji}$ matrix. Thus, the check node is updated.

$$\mathbf{L}(r_{ji}) = \log\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right). \qquad (8)$$

Thus

$$\mathbf{L}(r_{ji}) = \pi\alpha_{ij} \cdot \min \beta_{ij}. \qquad (9)$$

In this way, the algorithm of Min-Sum is derived from Log-SPA algorithm [18] by replacing the horizontal step by the above Eq. (9) [13].

1.3. Vertical Step

In this step, the variable node is updated. The $\mathbf{L}(q_{ij})$ is updated for next iteration and $L(Qi)$ is obtained for decoding purpose. From $\mathbf{L}(r_{ij})$ matrix, variable node is updated to obtain $\mathbf{L}q_{ij}$ matrix. The variable node is updated by adding all the elements (except the one being updated) in the column of $\mathbf{L}(r_{ji})$ matrix with the respective element from $\vec{L}(c_i)$. This is done for all elements of $\mathbf{L}r_{ij}$ matrix and thus variable node is updated.

$$L(Q_i) = \vec{L}(c_i) + \sum_{j \in C_i} \mathbf{L}(r_{ji}), \qquad (10)$$

$$\mathbf{L}(q_{ij}) = \vec{L}(c_i) + \sum_{j' \in C_i \setminus j} \mathbf{L}(r_{j'i}). \qquad (11)$$

1.4. Decoding / Estimation

$$\hat{C}_i = \begin{cases} 1 & \text{if} \quad L(Q_i) < 0, \\ 0 & \text{if} \quad L(Q_i) > 0. \end{cases} \qquad (12)$$

In the decoding step, the output vector is estimated depending on the value of $L(Qi)$. The decoded vector is verified by calculating syndrome.

1.5. Syndrome Calculation

The calculation of syndrome is done to verify the received data with the original data.

The syndrome is calculated by using the below formula:

$$S = \left(C_i \cdot \mathbf{H}^T\right) \bmod 2. \tag{13}$$

If the syndrome is not zero, next iteration has to be continued, until the syndrome is obtained as zero.

2. Optimized Min-Sum Decoding

Optimized Min-Sum decoding improves decoding procedure by introducing an optimization factor at check nodes. Consider an optimization factor as 'A'. To obtain the optimized Min-Sum algorithm, it is required to multiply the optimization factor 'A' in the horizontal step so that the number of iterations can be reduced compared to the Min-Sum algorithm.

From the horizontal step obtained in Eq. (9), it can be modified as:

$$\mathbf{L}(r_{ji}) = A \cdot \pi\alpha_{ij} \cdot \min \beta_{ij}. \tag{14}$$

Here, it is observed that before the decoding process, the optimization factor is determined which causes no additional complexity in the decoding algorithm. Thus, the algorithm proposed is a very efficient decoding technique in case of irregular LDPC codes [19]. Because of introduction of this optimization factor, there will be an additional multiplication in the horizontal step, which will be compensated by decrease in the number of iterations required for decoding the message. Matlab simulations are performed for various values of 'A' and value of 'A', for which the least Bit Error Rate (BER) is obtained, is chosen as the optimization factor. I.e. to find the optimization factor 'A', simulation is done for various values of 'A' in steps of 0.1, from 0 to 1 in the decoding algorithm. This is done for different values of SNR, as the value of the optimization factor varies with SNR and the value which produces the lowest BER is considered as the optimization factor for the given SNR. The simulation is taken for 100 different trials for one value of SNR [15].

3. Modified Min-Sum Decoding Algorithm

In the present work, modified Min-Sum algorithm is proposed, to further improve the performance by decreasing the number of additions and comparisons in the decoding process. In modified Min-Sum decoding algorithm, complexity is reduced in calculating the values of $\mathbf{L}q_{ij}$ required for further iterations. In Min-Sum algorithm, two different steps are used to calculate values of $\mathbf{L}q_{ij}$ and $L(Qi)$. This increases the delay in decoding the data. In modified Min-Sum algorithm, the number of additions required is reduced by including the process of subtraction to compute $\mathbf{L}q_{ij}$. This is due to the similarity in computing the values of $\mathbf{L}q_{ij}$ and $L(Qi)$. Modification is done only in the vertical step of the algorithm. Equation (10) and Eq. (11) are modified into the following equations:

$$L(Q_i) = \vec{L}(c_i) + \sum_{j \in C_i} \mathbf{L}(r_{ji}), \tag{15}$$

$$\mathbf{L}(q_{ij}) = L(Q_i) - \mathbf{L}(r_{ji}). \tag{16}$$

Normally LQ_i is computed by adding all values of $\mathbf{L}r_{ji}$ along with $\vec{L}c_i$. Then $\mathbf{L}q_{ij}$ is computed by updating each element of $\mathbf{L}r_{ij}$ by adding all the values of $\mathbf{L}r_{ij}$ except the value being updated. Since already the value of LQ_i is calculated, the value of each element in $\mathbf{L}q_{ij}$ is calculated using only one subtraction. Hence, number of additions drastically decreases.

4. MATLAB Simulation

The Min-Sum algorithm is implemented in MATLAB and its performance is measured. The block diagram of digital communication system is shown below.

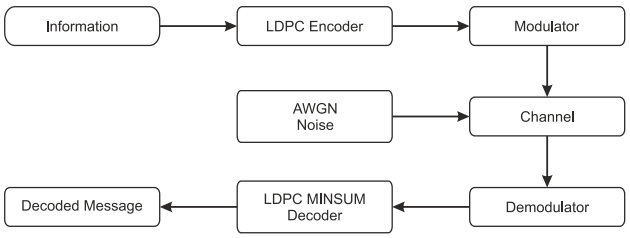

Fig. 1: Block Diagram of Digital Communication system.

The input message is given in decimal form and is converted to 4-bit binary values. Using the **H** matrix, generator matrix is obtained and the code vector [c] of 16-bits is generated by multiplying code vector with Generator Matrix. The coded bits are transmitted using BPSK modulation through AWGN channel. Noise gets added to the code vector and this channel output [r] is fed to the Min-Sum decoder, where the introduced errors are detected and removed obtaining decoded vector as d. The simulation result of the output of different steps of Min-Sum decoder is plotted for **H** matrix of dimensions (64, 96) as shown in Fig. 2.

The graphs for comparison of performance among Min-Sum, optimized Min-Sum and modified Min-Sum is plotted taking SNR values on the X-axis and BER values on the Y-axis. The graph is plotted for **H** matrix of dimension 8×12, 64×96 and 504×1008 [20] as

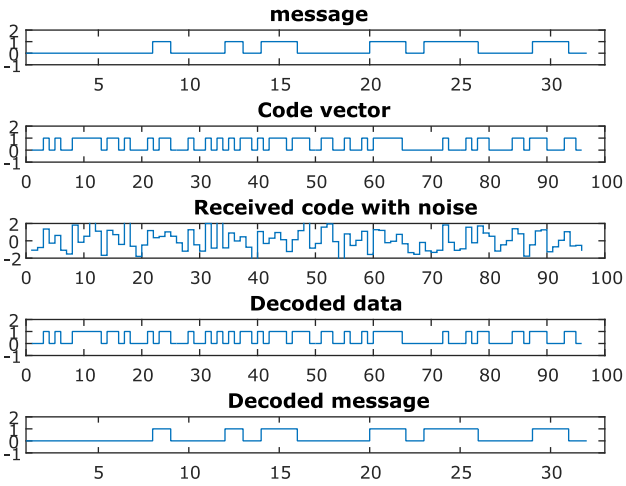

Fig. 2: Simulation results of Min-Sum Algorithm for **H** matrix of dimension 64×96.

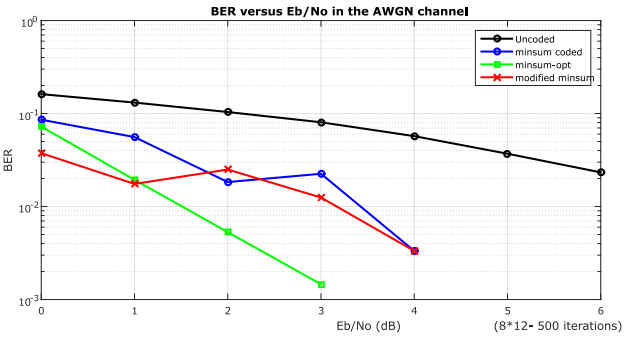

Fig. 3: BER performances for **H** matrix of dimension (8×12) for 500 iterations.

shown in Fig. 3 to Fig. 6. For plotting, SNR in increments of 1dB is considered, 1000 test bits for each SNR are sent, each decoding is set to the maximum of 500 iterations. The BER becomes zero for higher values of SNR. Good error performance is obtained even for low values of SNR. The code becomes more accurate if BER is lower for low values of SNR. It can be noticed that BER for optimized Min-Sum is lower than that of Min-Sum and modified Min-Sum decoder for a given value of SNR. To find the optimization factor, it is required to perform elaborate simulations and find the value for which BER is zero. These calculations may result in delay during computation. Hence, computing the optimization factor beforehand is necessary. As the matrix dimension increases, the decoding becomes more accurate and the BER becomes low for lower values of SNR. But if the size of **H** matrix increases, the computation time also increases. Thus, the selection of **H** matrix plays a crucial role in the decoding for a given message length. The simulation has also been run for a different number of iterations. Figure 6 shows the simulation result for 1000 iterations. It can be seen that when the number of iterations increases, the BER performance also improves.

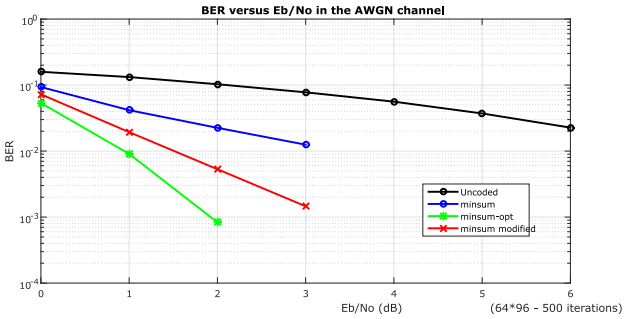

Fig. 4: BER performance for **H** matrix of dimension (64×96) for 500 iterations.

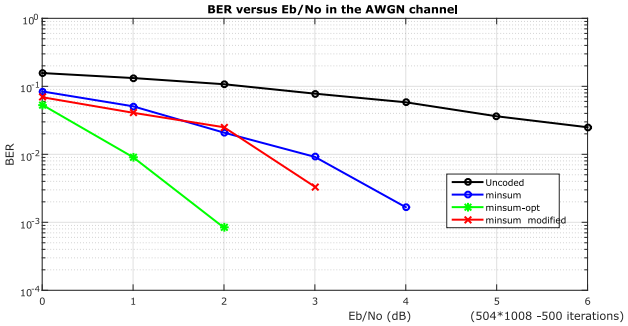

Fig. 5: BER performance for **H** matrix of dimension (504×1008) for 500 iterations.

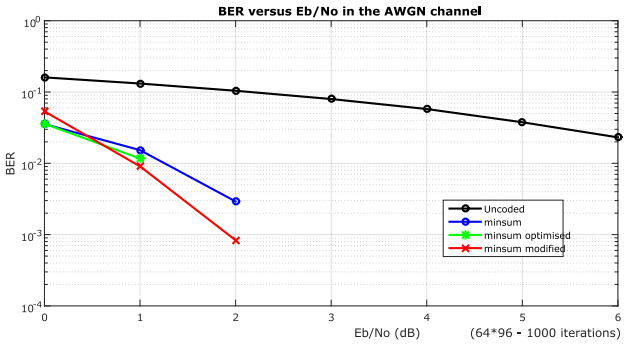

Fig. 6: BER performance for **H** matrix of dimension (64×96) for 1000 iterations.

5. FPGA Implementation

5.1. System Level Architecture

The Min-Sum algorithm uses RAM and ROM to store data. The Lc values from MATLAB are converted into fixed-point numbers and are stored in ROM-Lc in the form of Look Up Tables (LUT). The **H** matrix is also stored in ROM-**H** for the purpose of decoding. During initialization, the $\mathbf{L}q_{ij}$ is calculated and stored in RAM-L_q so that it can be updated later. The alpha and beta values are also stored in RAM-α and RAM-β, respectively. Similarly, the $\mathbf{L}r_{ij}$ is calculated and stored in RAM for further access during the horizontal

step. During the vertical step, L_{out} is calculated and stored in RAM-L_{out}. Lqij is also calculated and RAM-L_q is updated in this step. The L_{out} from this step is used for estimating the decoded vector. The Processor core is a Zedboard Zync-702 FPGA. The Min-Sum decoding algorithm is present within the FPGA processing core and its architecture is shown below.

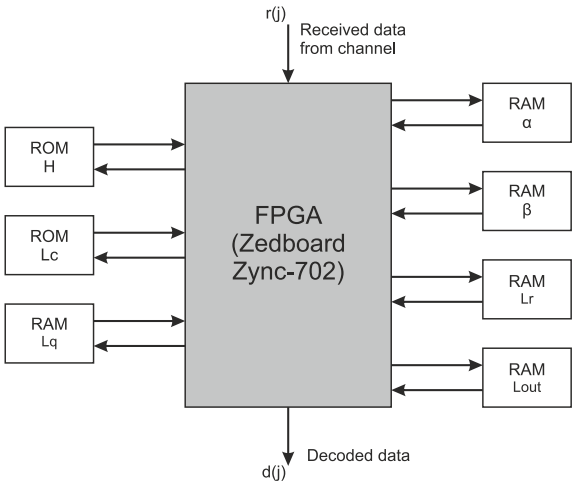

Fig. 7: Block diagram of Min-Sum Decoder in FPGA.

5.2. FPGA Architecture of Min-Sum Decoder

The Min-sum decoding algorithm is implemented on FPGA, and the algorithm is divided into 8 stages. The encoding and the addition of noise to the encoded data are done in MATLAB. The 8 stages are explained below.

Stage 1: In this step, input 'clock' is taken from the FPGA kit. The input $\vec{L}c_j$ values are given as input using a lookup table method. These values are obtained from Matlab. The values are converted into fixed-point numbers and are stored as 16-bit data in ROM. The 16-bit number has 4 bits before radix point and 12 bits after radix point. This gives the numbers an accuracy of '0.0002441'. The **H** matrix is stored in the form of lookup table in ROM. In the Initialization part, the $\vec{L}c_i$ values are placed in $\mathbf{L}q_{ij}$ when the corresponding elements in the **H** matrix are equal to one. Hence it converts $\vec{L}c_j$ vector of the length 'n' into $(n-k)\times n$ matrix $\mathbf{L}q_{ij}$.

Stage 2: In this step, we split $\mathbf{L}q_{ij}$ matrix into its magnitude matrix (β_{ij}) and sign matrix (α_{ij}). The MSB of each element is checked in the $\mathbf{L}q_{ij}$ matrix. If MSB is '1', then it is considered to be a negative number and hence the number is negated and stored in β. Also, α is set to -1. If MSB is zero, it is considered to be a positive number and hence the number is stored as it is in β. The α is made as '1'. Thus, both α and β are updated.

Stage 3: Now the value of $\beta_{\min}$ is set to the highest number that can be represented, i.e. 7FFF. The β values are compared in each row and the lowest β value is found out using bubble sort. This smallest value is assigned to $\beta_{\min}$. Then the signs of all these numbers are multiplied with each other to obtain product of signs. Then the sign is multiplied with $\beta_{\min}$ to obtain the $\mathbf{L}r_{ij}$ values of each position. This step is repeated for all values of $\mathbf{L}r_{ij}$.

Stage 4: From the $\mathbf{L}r_{ij}$ matrix, $\mathbf{L}q_{ij}$ values are obtained for the next iteration by adding all values of Lr_i except for the corresponding elements that need to be updated in each column. These steps are repeated until the complete $\mathbf{L}q_{ij}$ matrix is obtained.

Stage 5: After obtaining $\mathbf{L}r_{ij}$ matrix, the vertical step is initiated. In this step, a variable called 'sum' is used to store the temporary sum of addition. From $\mathbf{L}r_{ij}$, L_{out} is obtained by adding all the elements in each column to obtain a single sum per column. $\vec{L}c_j$ is added to this sum to obtain the L_{out}. These steps are repeated to obtain the entire $\vec{L}_{out}$ vector. The $\vec{L}_{out}$ will be a vector of length 12.

Stage 6: From the $\vec{L}_{out}$ vector, the decoded vector is obtained. The MSB of $\vec{L}_{out}$ of each element is checked. If it is 0, the number is positive, and output is made d=0. If MSB is 1, the number is negative, and output is made d=1.

Stage 7: In this stage, the transpose of **H** is got by interchanging the i and j variables of **H** matrix; $(n-k)\times n$ matrix is converted into $n\times(n-k)$.

Stage 8: In this stage, a temporary variable called 'temp' is used to calculate syndrome. First, AND operation is performed between output data and **H** transpose. Then, XOR between this result and temp is performed. Then, the syndrome is updated with value of temp.

Stage 9: In this stage, comparison is done to check, if the syndrome is 0 using a comparator. If the syndrome is zero, the output is placed in 'd' and then computation is stopped. If the syndrome is not zero, then go back to Stage 2 and perform all the operations again. The updated $\mathbf{L}q_{ij}$ is used for upcoming iterations.

5.3. FPGA Implementation of Optimised Min-Sum Decoding

In the optimized Min-Sum Decoder, a new factor called an optimization factor is introduced in the horizontal step of the decoding algorithm which helps in increasing the accuracy of the algorithm. The optimisation factor is multiplied to update $\mathbf{L}r_{ji}$ in the horizontal

step which will normalise the value of $\mathbf{L}r_{ji}$. Thus optimisation factor is used to update the check node. The Eq. (9) is updated to:

$$\mathbf{L}(r_{ji}) = A \cdot \pi\alpha_{ij} \cdot \min(\beta_{ij}), \tag{17}$$

where 'A' is the optimization factor.

Since an extra variable is used to store the value of the optimization factor, the number of LUT required will increase. But the accuracy of the design also increases and hence the time in which the decoding completes decreases. This optimization is done in Stage 3 of FPGA design shown in Fig. 8 and it updates only the check node. It does not affect the variable node. Thus the complexity of the algorithm does not increase.

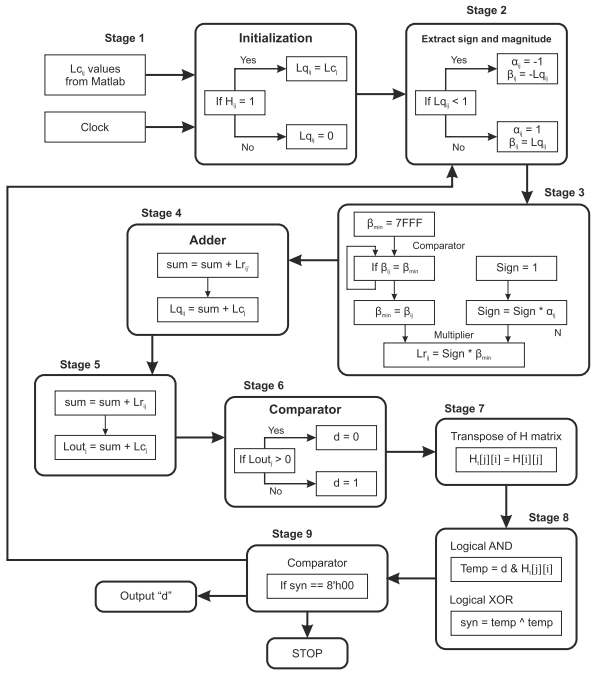

Fig. 8: FPGA architecture of Min-Sum Decoder.

5.4. FPGA Implementation of Modified Min-Sum Decoding Algorithm

Modified Min-Sum algorithm is proposed to further improve the performance by decreasing the number of stages in the decoding process in FPGA implementation. In modified Min-Sum decoding algorithm, complexity is reduced in calculating the values of $\mathbf{L}q_{ij}$ required for further iterations. In Min-Sum algorithm, 2 different stages are used to calculate values of $\mathbf{L}q_{ij}$ and LQ_i. This increases the delay in decoding the data. In modified Min-Sum algorithm, both of these values are calculated in a single stage. Thus the number of additions decreases. This is due to the similarity in computing the values of $\mathbf{L}q_{ij}$ and LQ_i . Modification is done only in the vertical step of the algorithm. The formulae for these values is given by Eq. (10) and Eq. (11). These equations are modified into the following equations shown below.

$$L(Q_i) = \vec{L}(c_i) + \sum_{j \in C_i} \mathbf{L}(r_{ji}), \tag{18}$$

$$\mathbf{L}(q_{ij}) = L(Q_i) - \mathbf{L}(r_{ji}). \tag{19}$$

Normally LQ_i is computed by adding all values of $\mathbf{L}r_{ji}$ along with $\vec{L}c_i$. Then $\mathbf{L}q_{ij}$ is computed by updating each element of $\mathbf{L}r_{ij}$ by adding all the values of $\mathbf{L}r_{ij}$ except the value being updated. But in modified Min-Sum decoding, the above two steps are implemented in a single stage as indicated in Fig. 9, which shows the stage 4 in FPGA architecture mentioned in Fig. 8. Since the value of LQ_i is already calculated, the value of each element in $\mathbf{L}q_{ij}$ is calculated using only one subtraction. Hence, number of additions drastically increases. Moreover, since the values are computed in a single stage, the number of nodes to be updated decreases thus increasing the operating frequency of the overall decoding algorithm. The usage of memory and LUT increases by a small amount, since computation of these values is done in a single stage. Thus there is a trade-off between operating frequency and memory utilisation. But the increase in operating frequency overweighs the increase in memory thus making this algorithm more suitable for practical implementation.

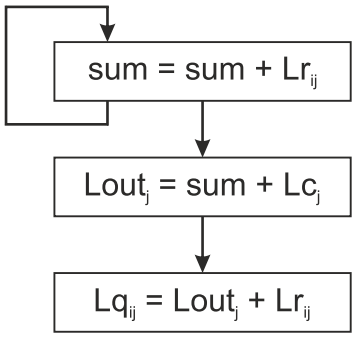

Fig. 9: FPGA architecture of Modified Min-Sum Algorithm.

5.5. Synthesis Result Comparison between Min-Sum, Optimised Min-Sum and Modified Min-Sum Decoder

Product Family: Zync-702.
Product Part: ZedBoardZynq Evaluation and Development Kit (xc7z020clg484-1).
Device: xc7z020.
Package: clg484.

Tab. 1: Comparison between Min-Sum, Optimised Min-Sum and Modified Min-Sum decoder.

Parameters	Min-Sum	Optimised Min-Sum	Modified Min-Sum
LUT	14976	17749	15740
LUTRAM	6	6	8
FF	6293	6199	6216
DSP	2	2	2
Modified Minimum Period	42 ns	40.5 ns	39 ns
Maximum Frequency	23.81 MHz	24.691 MHz	25.641 MHz
Slack	2.924 ns	6.240 ns	2.455 ns
Total On-Chip Power	0.164 W	0.18 W	0.188 W

Min-Sum Decoder uses less LUTs because it does not have any extra variables to store. The operating frequency is lower than other techniques for this Decoder because it consists of an extra stage for decoding. Optimized Min-Sum Decoder, on the other hand, uses a variable called an optimization factor which is used for check node update. This optimization factor needs a register to be stored and also to be updated frequently. Hence the number of LUTs increases. But the frequency increases since the optimization factor increases the accuracy and also decreases the number of iterations required for decoding. The value of Slack in Optimized Min-Sum Decoder is higher because the time delay between different stages is longer due to the introduction of the optimization factor. The Modified Min-Sum Decoder balances between the use of LUTs and also operating frequency. This decreases one stage in FPGA implementation and thus increases the frequency of operation of the code. The increase in LUT over that of Min-Sum Decoder is because of the fact that two stages are combined into one stage which increases memory usage.

6. Conclusion

In optimized Min-sum Decoder, the optimization factor is introduced in the horizontal step, so that the number of iterations for decoding decreases. The computational requirement for the optimization factor is high. Hence, a Modified Min-Sum LDPC Decoder is proposed for the efficient operation which can detect and correct errors with less decoding time and further decreases one stage in FPGA implementation and thereby increases the frequency of operation of the code. Thus, Modified Min-Sum Decoder increases the operating frequency without much increase in the resource utilization. The graphs showing the BER performance of Min-Sum, Optimized and Modified Min-Sum algorithm are plotted for parity check matrix combinations such as (8×12), (64×96) and (504×1008). FPGA implementation is done for Min-Sum, Optimized and Modified Min-Sum algorithm for parity check matrix (8×12) using Zynq-702 boards and the results are obtained in VIVADO. The algorithm can be further implemented for the higher parity check matrix on the hardware.

References

[1] SHANNON, C. E. A Mathematical Theory of Communication. *Bell System Technical Journal*. 1948, vol. 27, iss. 3, pp. 379–423. ISSN 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[2] GALLAGER, R. Low-density parity-check codes. *IEEE Transactions on Information Theory*. 1962, vol. 8, iss. 1, pp. 21–28. ISSN 0018-9448. DOI: 10.1109/TIT.1962.1057683.

[3] BERROU, C., A. GLAVIEUX and P. THITIMAJSHIMA. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In: *Proceedings of ICC '93 - IEEE International Conference on Communications*. Geneva: IEEE, 1993, pp. 1064–1070. ISBN 0-7803-0950-2. DOI: 10.1109/ICC.1993.397441.

[4] MACKAY, D. J. C. and R. M. NEAL. Near Shannon limit performance of low density parity check codes. *Electronics Letters*. 1997, vol. 33, iss. 6, pp. 457–458. ISSN 0013-5194. DOI: 10.1049/el:19970362.

[5] MACKAY, D. J. C. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*. 1999, vol. 45, iss. 2, pp. 399–431. ISSN 0018-9448. DOI: 10.1109/18.748992.

[6] MOHSENIN, T. and B. BAAS. Trends and challenges in LDPC hardware decoders. In: *Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*. Pacific Grove: IEEE, 2009, pp. 1273–1277. ISBN 978-1-4244-5825-7. DOI: 10.1109/ACSSC.2009.5469947.

[7] MORELLO, A. and V. MIGNONE. DVB-S2: The Second Generation Standard for Satellite Broad-Band Services. *Proceedings of the IEEE*. 2006, vol. 94, iss. 1, pp. 210–227. ISSN 0018-9219. DOI: 10.1109/JPROC.2005.861013.

[8] CASTINEIRA, M. J. and P. G. FARRELL. *Essentials of Error-Control Coding*. Chichester: John Wiley & Sons, 2006. ISBN 978-0-4700-3572-6.

[9] KSCHISCHANG, F. R., B. J. FREY and H. A. LOELIGER. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*. 1998, vol. 47, iss. 2, pp. 498–519. ISSN 0018-9448. DOI: 10.1109/18.910572.

[10] ZHANG, T. and K. K. PARHI. An FPGA Implementation of-Regular Low-Density Parity-Check Code Decoder. *EURASIP Journal on Advances in Signal Processing*. 2003, vol. 2003, iss. 6, pp. 530–542. ISSN 1687-6180. DOI: 10.1155/S1110865703212105.

[11] SADEK, A. M. and A. I. HUSSEIN. Flexible FPGA implementation of Min-Sum decoding algorithm for regular LDPC codes. In: *11th International Conference on Computer Engineering & Systems*. Cairo: IEEE, 2016, pp. 286–292. ISBN 978-1-5090-3267-9. DOI: 10.1109/ICCES.2016.7822016.

[12] WIBERG, N. *Codes and Decoding on General Graphs*. Linkoping, 1996. Dissertation. Linkoping University. Supervisor prof. Ingemar Ingemarsson.

[13] RYAN, W. *An Introduction to LDPC Codes*. Boca Raton: CRC Press, 2004. ISBN 978-0-8493-1524-4.

[14] HAN, W., J. HUANG and F. WU. A modified Min-Sum algorithm for low-density parity-check codes. In: *IEEE International Conference on Wireless Communications, Networking and Information Security*. Beijing: IEEE, 2010, pp. 449–451. ISBN 978-1-4244-5850-9. DOI: 10.1109/WCINS.2010.5541818.

[15] RAKIBUL, M., D. SIAM, M. MOSTAFA and I. RAHMAN. Optimized Min-Sum Decoding Algorithm for Low Density Parity Check Codes. *International Journal of Advanced Computer Science and Applications*. 2011, vol. 2, iss. 12, pp. 168–174. ISSN 2158-107X. DOI: 10.14569/IJACSA.2011.021225.

[16] WU, X., Y. SONG, L. CUI, M. JIANG and C. ZHAO. Adaptive-normalized min-sum algorithm. In: *2nd International Conference on Future Computer and Communication*. Wuha: IEEE, 2010, pp. 661–663. ISBN 978-1-4244-5821-9. DOI: 10.1109/ICFCC.2010.5497569.

[17] SAVIN, V. Self-corrected Min-Sum decoding of LDPC codes. In: *IEEE International Symposium on Information Theory*. Toronto: IEEE, 2008, pp. 146–150. ISBN 978-1-4244-2256-2. DOI: 10.1109/ISIT.2008.4594965.

[18] ARNONE, L., C. GAYOSO, C. GONZALEZ and J. CASTINEIRA. Sum-subtract fixed point LDPC decoder. *Latin American Applied Research*. 2006, vol. 37, iss. 1, pp. 17–20. ISSN 0327-0793.

[19] ZHANG, J., M. FOSSORIER, D. GU and J. ZHANG. Two-dimensional correction for min-sum decoding of irregular LDPC codes. *IEEE Communications Letters*. 2006, vol. 10, iss. 3, pp. 180–182. ISSN 1089-7798. DOI: 10.1109/LCOMM.2006.1603377.

[20] MACKAY, D. J. C. Online database of low-density parity check codes. *Creative Artists Management* [online]. 2018. Available at: `http://wol.ra.phy.cam.uk/mackay/codes/data.html`.

[21] GHAFFARI, F., B. UNAL, A. AKOGLU, K. LE, D. DECLERCQ and B. VASIC. Efficient FPGA implementation of probabilistic gallager B LDPC decoder. In: *24th IEEE International Conference on Electronics, Circuits and Systems*. Batumi: IEEE, 2017, pp. 178–181. ISBN 978-1-5386-1911-7. DOI: 10.1109/ICECS.2017.8292048.

[22] APEKSHA, Y., S. KAKDE, A. KHOBRAGADE, D. BHOYAR and S. KAMBLE. LDPC Decoder's Error Performance over AWGN Channel using Min-Sum Algorithm. *International Journal of pure and applied mathematics*. 2018, vol. 118, iss. 20, pp. 3875–3878. ISSN 1314-3395.

[23] SREEMOHAN, P. V. and N. SEBASTIAN. FPGA implementation of min-sum algorithm for LDPC decoder. In: *International Conference on Trends in Electronics and Informatics*. Tirunelveli: IEEE, 2017, pp. 821–826. ISBN 978-1-5090-4257-9. DOI: 10.1109/ICOEI.2017.8300818.

About Authors

Rajagopal ANANTHARAMAN was born in 1977 in Tumkur, Karnataka, India. He has received M.Tech degree in VLSI Design & Embedded systems from VTU in 2006. He is presently working as an Asst. Prof. in Dayananda Sagar College of Engineering, Bangalore. He is currently working towards his Ph.D. degree in VTU. His research interest includes Digital System Design, Digital Communication systems, Error Control Coding, FPGA System Design, VLSI design and Embedded systems.

Karibasappa KWADIKI was born in 1962 in Bellary, Karnataka, India is presently working as Professor in DSATM, Bangalore. He has received his Ph.D. degree in Machine Learning and Perception Using Cognitive Methods in 2004. His Research interest includes Machine Learning and Perception, Artificial Intelligence, Data Mining, Knowledge Acquisition, Digital Image Processing and Error Control coding.

Vasundhara Patel KEREHALLI SHANKAR RAO was born in 1965 in Shimoga, Karnataka, India. She is presently working as Professor in BMSCE, Bangalore. She has received her Ph.D. degree in Electronics Engg. In 2012. Her domain of Interest includes Multi-Valued Logic Design, Nano Electronics, Low power VLSI, Analog and Mixed signal VLSI Design, Digital VLSI Circuit Design, Physical Design.