Al-Rafidain Engineering Journal (AREJ) Vol. 29, No. 2, September 2024, pp. 56-64

Image Transmission Through Channel Coding Architecture

Zainab Ali Alkhayat
zainab.21enp6@student.uomosul.edu.iq

Dhafir A. Alneema
dhafir.abdulfattah@uomosul.edu.iq

Computer Engineering Department, College of Engineering, University of Mosul, Mosul, Iraq

Received: January 14th, 2024 Received in revised form: Febuary 13th, 2024 Accepted: March 4th, 2024

ABSTRACT

Image transmission in modern communication systems needs fast and low error coding and imperative transmission mechanisms. For engineers, dependable communication over a noisy channel is a long-standing but difficult problem. One of the important types of channel coding is Low-Density Parity-Check (LDPC) codes which are considered Linear Block Codes (LBC). Due to their superior error-correcting ability, LDPC codes are among the most widely used Forward Error Correction (FEC) codes. The purpose of this paper is to create transmitter channel encoder LDPC architectures and the corresponding channel decoder in the receiver for image transmission. The study integrates the Effective Encoding of the LDPC codes algorithm for the encoder and the Bit flipping LDPC codes algorithm for the decoder, Vivado HLS (High_Level Synthesis) is the tool utilized in this work, The HLS loop unrolling optimizing technique is used to give the synthesizer instructions on how to implement a particular code section, the designer can quickly and easily optimize the application, as a result, optimization is done directly on the source code. Additionally, it suggests applying optimization techniques like loop unrolling to every design. C programming language and HLS are used to create all architectures.

Keywords:

Image transmission; Effective Encoding of LDPC; Bit flipping LDPC; Vivado HLS.

https://rengj.mosuljournals.com
Email: alrafidain_engjournal2@uomosul.edu.iq

==========

1. INTRODUCTION

Image transmission refers to the process of sending images from one location to another. This makes long-distance visual information sharing possible, opening up new possibilities for applications in communication, remote sensing, healthcare, and entertainment. Communication is considered one of the key domains where picture transmission is essential. Real-time image transmission enhances the richness and context of interactions in various communication contexts, from video conferencing platforms to personal messaging apps, and makes information sharing more effective. This technology has reinforced social media platforms, enabling users to visually share their experiences with friends, family, and the global community.

LDPC codes have drawn the attention of researchers and are now an essential area of study. The study of [1] summarizes the main characteristics and advantages of the LDPC codes, which are as follows:

a- Fast decoding in addition to good error-performance close to Shannon capacity [2].
b- Basic decoding techniques, such as Message-Passing decoding.
c- The size of the blocks increases the decoding's complexity.
d- Permit simultaneous execution.
e- Flexibility in parameter selection.
f- Capable of delivering an outstanding performance in contemporary mobile communication systems.

The necessary steps for designing and implementing image transmission through the LDPC are for the encoder using an efficient method and for the corresponding decoder using a bit-flipping algorithm to recover the original image. Figure 1 shows the block diagram of image transmission through channel coding. The image passes through pre-processing steps. All the design architectures were done using Vivado HLS with a C programming language. The Xilinx Vivado

Design Suite Package release 2018.3 was used for synthesizing the architectures.

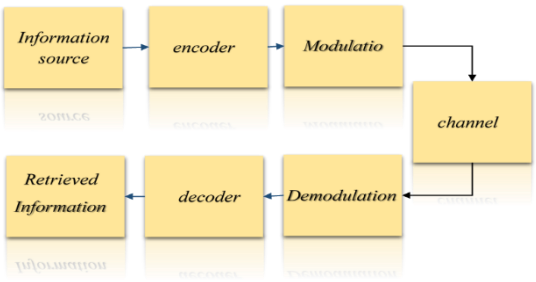


Figure 1: The general diagram of the Digital Communication System

The key characteristics of LDPC Codes [3] are that they are capable of providing efficient encoding and decoding with reduced decoding time, latency, and error-floors at high SNR and low BER for low SNR [4, 5]. LDPC codes have better error detection and correction capabilities. LDPC codes can be used to add parity bits to a message before sending it to the receiver, allowing the receiver to determine what message the sender wants to send [6].

The FPGAs' implementation offers a good balance between developme8nt flexibility, algorithm testing and reconfigurability, real-time performance, and costs [7-9]. Generally, a system on chip (SoC) enables the integration of current hardware (HW) acceleration and software (SW) libraries into a single, small device. As a result, this technique enables reductions in both size and power usage [10, 11].

The literature review on encoders, LDPC decoders, and image transitions (encoders and decoders) is presented in this section along with suggestions for improvement. The literature has examined a wide range of topics to produce ideal designs that can be successfully implemented in the manufacturing sector. In the field of creating LDPC decoders, there are numerous surveys and reviews in the literature. This section's primary objective is to highlight the studies that are more closely relevant to this paper's subject.

The study of [12] gave a literature review on different kinds of image algorithms and associated theories. Many quality criteria have been debated to do a brief comparison of these methods. The paper [13] described the 3D video coding using FPGA encoder architecture for more recent and dependable multimedia technologies. The goal is to push the industry to enhance services in the field of entertainment marketing, to promote the uptake of 3D video content, and to support 3D devices and applications.

In the study [14], the suggested system presented a simplified model that requires a thorough channel encoding to transmit data effectively. The suggested system channel encoding performs consistently across a range of image types and dimensions. The entire encoding scheme is based on the packetization concept, in which the communication bits are created according to the traffic load at any given time to provide greater transmission flexibility. Additionally, the system has implemented an efficient indexing mechanism that drastically reduces the transmitted image size while having no impact whatsoever on the reconstructed image signal quality.

Nadzri, Muhamad, and Afandi proposed in the study [15] that the image input data in the wildlife surveillance system to monitor the wildlife. This study proposes a SoC FPGA rapid prototyping system architecture for efficient image transmission for the wildlife surveillance system. It offers higher integration, lower power consumption, smaller board size, and higher bandwidth communication between the processor and the FPGA platform. The prototype image compression system on the FPGA platform using the DCT technique was able to decrease image size and transmission time, increasing system efficiency. A reduced transmission time was achieved by successfully reducing the image's size. In addition, the use of AES-based image data encryption was established in the proposed prototype. It was ensured by employing the AES technology that the supplied image data would only be viewed and read if the password was provided correctly.

A high-level synthesis tool was used in [16] for implementing a turbo decoder based on one map algorithm with the use of some HLS optimization directives for improving the design. In the study of [17], the researchers made a comparison between LDPC and TURBO codes based on various parameters like latency, speed, and efficiency. The Bit Error Rate Ratio (BER) was assessed as their values. Accordingly, the performance and complexity of both types of codes were analyzed and evaluated. LDPC codes had a higher decoding complexity compared to TURBO codes. Nevertheless, LDPC codes are more efficient than TURBO codes. It was also observed that LDPC code was executed faster than turbo code. Moreover, compared to TURBO codes, LDPC had a lower BER ratio.

2. THE THEORETICAL BASIS

The following is the theoretical basis for Effective Encoding and bit flipping for decoding of LDPC:

2.1 Effective Encoding Steps of LDPC

The preprocessing technique described for determining a generator matrix G for a given H can be applied to encode a vector of size (1 * m) of any arbitrary message bits.

Step 1: by performing row and column permutations, the non-singular parity check matrix H is to be brought into a lower triangular form, as indicated in Figure 2. More precisely, the H matrix is brought into the form:

$$H^t = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \quad (1)$$

with a gap g as small as possible. Where A is (m - g) × (n - m) matrix, B is (m - g) × g matrix, T is (m - g) × (m - g) matrix, C is g × (n - m) matrix, D is g × g matrix and E is g × (m - g) matrix. All of these matrices are sparse and T is lower triangular with ones along the diagonal.

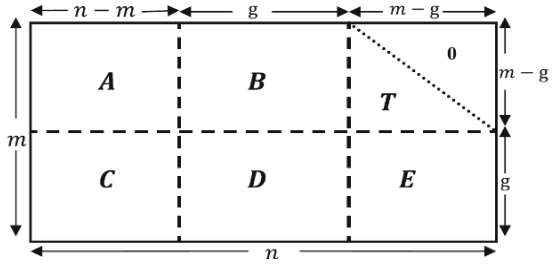


Figure 2: The parity check matrix in approximate lower triangular form

Step 2: premultiply H^t by

$$\begin{bmatrix} I_{m-g} & 0 \\ -ET^{-1} & I_g \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} I_{m-g} & 0 \\ -ET^{-1} & I_g \end{bmatrix} \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} =$$

$$\begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \quad (3)$$

To verify that $(-ET^{-1}B + D)$ is not a singular. It needs to be confirmed by running additional column permutations.

Step 3: utilizing the following, obtain P_1.

$$P_1^T = -\emptyset^{-1}\,(-ET^{-1}A + C)\,S^T \quad (4)$$

Where:
$-\emptyset^{-1} = (-ET^{-1}B + D)$ and S is the message vector.

Step 4: Utilize the following to obtain p_2.

$$P_2^T = -T^{-1}\,(As^T + BP_1^T) \quad (5)$$

Step 5: Create the c code vector as

$$C = [S\ P_1\ P_2] \quad (6)$$

The first g parity bit is stored in P_1, and the remaining parity bits are stored in P_2.

2.2 Bit-Flip Decoding Steps of LDPC

One kind of LDPC decoding algorithm is the bit-flipping (BF) method. Although it does not perform as well as the Min Sum algorithm at correcting errors, its low complexity allows for rapid implementation [18]. Hard-Decision Message Passing is a BF algorithm. This algorithm's primary process involves flipping the bits that are thought to be erroneous.

Until the codeword passes each parity check, the process is repeated. The check equations nearly always classify a failed bit in the codeword as an error bit. The matrix [H] is designed to be sparse to ensure that Parity-Check Equations are not focused on the same group of bits.

Gallager [2] made the initial suggestion for the algorithm, and the algorithm's steps of the BF can be illustrated as follows:

Step 1: Use ($r*H^T = S$) to calculate the syndrome's value, where r denotes the received bits. The decoding process is terminated if all of the parity checksums' values equal zero.

Step 2: The number of parity-check equation failures, for every node, is calculated. Next, for every message node, the number of failed check-nodes is computed.

Step 3: In this step, the S set of variable-node is located.

Step 4: The S's bits are then reversed.

Step 5: Repeat Steps 1 through 4 once more. When parity checksums give 0, which indicates that the decoding process is successful, the stop condition is held.

Moreover, all of the syndrome values equal zero in the absence of an error. In the same context, an error is identified when any one of the syndrome's values (s1, s2,..., sj) equals 1. In practice, the parity-checksums are calculated repeatedly by the decoder until all of their values are zeros [17].

Example: For the following parity check matrix,

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

let the received information be v=001000 as shown in Figure 3. From the syndrome calculations, we get: $S= v*H^T$ =1 0 0 1, which is not zero, If S = 0 0 0 0, it means no error, but in this case must correct the error, so this is not a valid codeword. The nodes 1 and 4 are the parity checks that have failed. This indicates that one or more of the symbols linked to the Tanner graph's check nodes, 1 and 4, contain an error. Since bit 4 of the

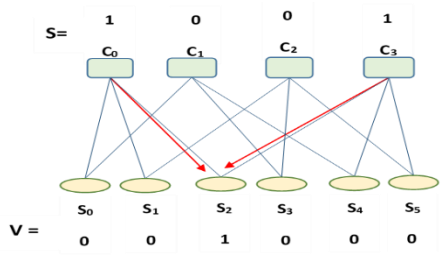


Figure 3: Selecting the error location from Tanner graph [six columns and four rows]

received vector is connected to check nodes 2 and 3, both of which are zeros in the syndrome vector, it indicates that there were no failed checks. As bits 1 and 2 of the received vector are connected to check node 1, they represent one failed check. Due to the fact that bits 5 and 6 of the received vector are connected to check node 4, they indicate that one check failed. As bit 3 of the received vector is connected to check nodes 1 and 4, which are both ones in the syndrome vector, it corresponds to two failed checks. We flip the 3rd bit according to the bit-flipping algorithm. Hence the correct received vector is 000000 [20].

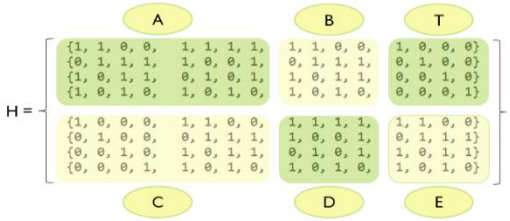
2.3 Guidelines for Optimization

Several design optimizations can be achieved by using Vivado-HLS such as task-guiding through using a "Pipeline" and defining a "Latency" limit for function completion, regions, loops, and resource usage. Override the operations that have inherent or implied dependencies.

Loop unrolling: usually, some unrolling of the loop is done. Think about the high-level codes for the loop. Every time it loops a certain amount of code is run in the body of the loop, the loop body executes instructions for multiple iterations and the number of iterations is reduced by unrolling the loop. As an illustration is considered a shift register, a loop is used to model it, with each register obtaining its value from the previous one. One register per cycle, however, is too slow to propagate the data. As a result, after the loop is fully unrolled, the value of the previous cycle is assigned to every register in turn. That is the shift register operating as it should.

3. LDPC Effective Encoder and BF Decoder HLS Implementation

When using the H matrix as follows:



All the steps of the encoder side including the image reading preprocessing are illustrated in the following flowchart Figure 4.

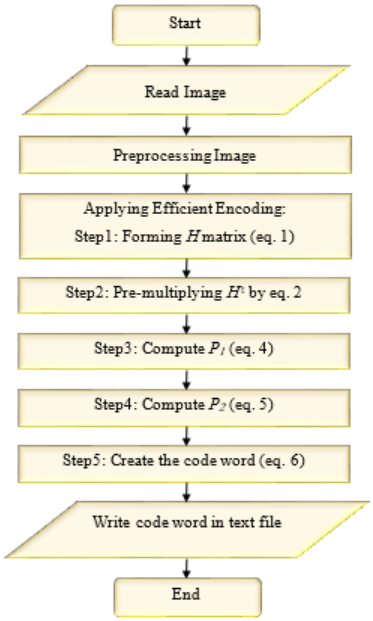


Figure 4: Encoder Core Flowchart

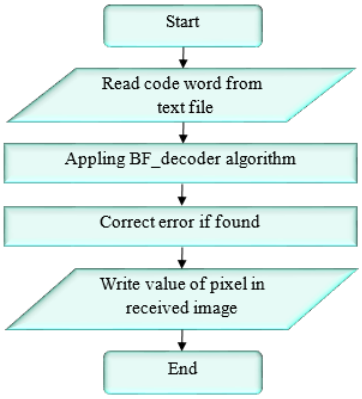


Figure 5: Decoder Core Flowchart

The flowchart Figure 5 illustrates all the steps of the decoder side applying the bit flipping algorithm and recovering the image.

The non-singular parity check matrix H is to be brought into a lower triangular form by executing row and column permutations, as shown by the H matrix being brought into the form:

3.1 Efficient LDPC Encoder Algorithm

The following points represent an example of constructing the H matrix. Where:
1- All these above matrices are sparse, and T is a lower triangular matrix with ones along the diagonal, we chose the T matrix as an identity to ease calculations and reduce resources and the run

time because T is an identity matrix equal to its inverse [21].

2- The following matrix was calculated by hand and entered as a binary matrix because it is constant in all cases to shorten the execution time and the resources used.

$$-\emptyset^{-1}\left(-ET^{-1}A+C\right)=\begin{matrix}\{1,\ 1,\ 1,\ 1,\quad 1,\ 0,\ 1,\ 1\}\\ \{0,\ 0,\ 1,\ 1,\quad 1,\ 0,\ 1,\ 0\}\\ \{0,\ 1,\ 0,\ 0,\quad 0,\ 0,\ 0,\ 1\}\\ \{1,\ 0,\ 1,\ 0,\quad 0,\ 0,\ 0,\ 1\}\end{matrix}$$

3- Obtaining P_1 using the following equation:

$$\begin{matrix}\{1,\ 1,\ 1,\ 1,\quad 1,\ 0,\ 1,\ 1\}\\ \{0,\ 0,\ 1,\ 1,\quad 1,\ 0,\ 1,\ 0\}\\ \{0,\ 1,\ 0,\ 0,\quad 0,\ 0,\ 0,\ 1\}\\ \{1,\ 0,\ 1,\ 0,\quad 0,\ 0,\ 0,\ 1\}\end{matrix}\ \mathbf{X}\ \begin{matrix}\{1\}\\ \{0\}\\ \{1\}\\ \{0\}\\ \{1\}\\ \{1\}\\ \{1\}\\ \{1\}\end{matrix}\ \Rightarrow\ \begin{matrix}\{1\}\\ \{1\}\\ \{1\}\\ \{1\}\end{matrix}$$

$$P_1^T=-\emptyset^{-1}\left(-ET^{-1}A+C\right)S^T$$

4- Then, obtain P_2 using the following equation

$$P_2^T=-T^{-1}\left(As^T+Bp_1^T\right)=\begin{matrix}\{1\}\\ \{0\}\\ \{1\}\\ \{0\}\end{matrix}$$

The codeword is formed as C ={ m p1 p2} and the result of C after running the program in HLS is shown in Figure 6:

C = {1010 1111 1111 1010 }

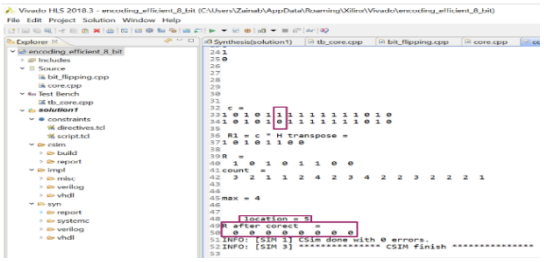


Figure 6: Encoder the value of (175)

3.2 -Bit flipping decoder algorithm

At the beginning, we should multiply the received codeword by the parity check matrix if the result equals (0) this means we receive the correct value and go to step 8. But if the result of the multiplication does not equal zero, it means that it has an error and must be corrected through the bit-flipping algorithm.

In each of the variable nodes calculate how many ones come from the syndrome through the link which is connected from this node.

The node that has a large number of ones in the variable node, represents the location of the error bit in the received vector.

After finding the location of the error bit, this bit should be flipped.

Finally, multiply the new received vector (after flipping) by the parity check matrix to check the error correction warranty.

Figure 7 shows the flowchart of the bit-flipping decoder algorithm that is in use.

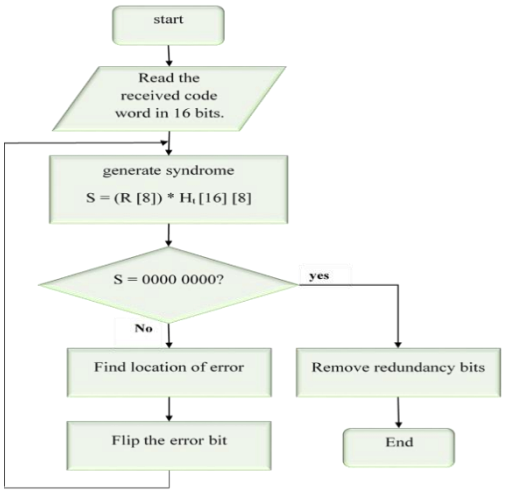


Figure 7: Bit Flipping Decoder Algorithm Flowchart

The received codeword C=(1010 1011 1111 11010) was multiplied by H_t and obtained the result of S= (10101100). Then applying the BF-decoder algorithm where the founding four ones' came to the location numbered 5th (starting location with 0th), this means its location has an error bit.

The results of the running of this example are demonstrated in Figure 8.

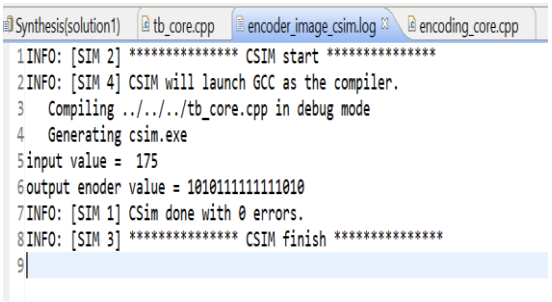


Figure 8: Finding the location of the error and correcting it

4. RESULTS AND DISCUSSIONS

The Design-Suite Package of Vivado was used to execute the encoder's operations. Additionally, the optimization was put into practice.

The image reading and writing in the Vivado HLS program need carefully formulated especially putting the image in an integer matrix because when we read it using the (imread)

function in OpenCV, the data will be stored in structure form and take many attributes and not easy to make some process. To overcome this problem, the data of the image must be put in an integer matrix, and then on the receiver side, the final step (after decoding all data) will be recovered in structure format (mat type).

To encode the 8-bit message, the example of one pixel in gray-scale M= 175 in binary representation m=[1010 1111], the resulting formula of the code $c = [m\ P_1^T\ P_2^T]$.

Then, after encoding, $P_1^T = [1111]$ and $P_2^T = [1010]$, so the codeword will be $c = [1010\ 1111\ 1111\ 1010]$. For checking, multiply this result by H^T and the result must be equal to zero to ensure that the encoder process walked in the true path.

Table 1 shows some of the pixel values and their conversion to the binary values and their corresponding encoded code.

Table 1: The values of some pixels and the corresponding codewords

Pixel Value in Decimal	Pixel Value in binary	Code Word
0	0000 0000	0000 0000 0000 0000
1	0000 0001	0000 0001 1011 0100
2	0000 0010	0000 0010 1100 1110
…	…	…
50	0011 0010	0011 0010 1101 1001
…	…	…
175	1010 1111	1010 1111 1111 1010
…	…	…
255	1111 1111	1111 1111 1001 1011

A-Synthesis results encoder in Vivado_HLS program for image (16*16)

The synthesis results of encoding the image (16*16) using and without using unrolling directives are shown in Figure 9. Figure 10 demonstrates the comparisons between different resources of an encoder of image (16*16) with and without using optimization.

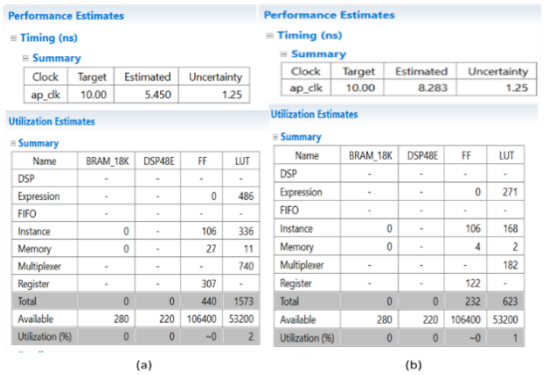


Figure 9: Effective LDPC encoding synthesis results (a) Without and (b) With using optimization technique for image (16*16)

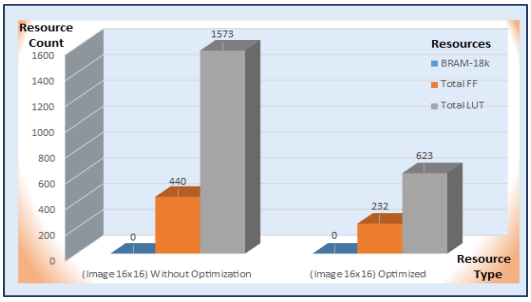


Figure 10: Resources of Encoder of Image (16*16)

Table 2 below provides the utilization report of the encoder design in Vivado HLS.

Table 2: The design report of the utilization of the encoder

Image size	16 * 16	
Resources	Without Optimization	Optimized
Target Time -Uncertainty	8.75 ns	8.75 ns
Timing Estimated	5.450	8.283
BRAM-18k	0	0
Total FF	440	232
Total LUT	1573	623
LUT in Multiplexer	740	182
Total Bits in Multiplexer	438	203
FF in Register	307	122
Const Bits in Register	130	0

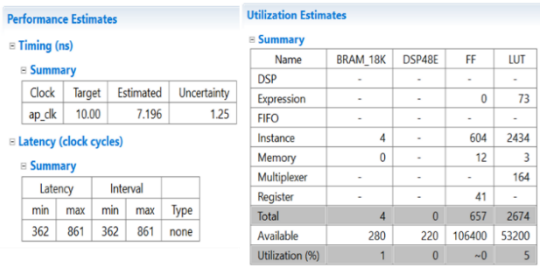


Figure 11: Synthesis running a report for applying BF-decoder without optimization with (16*16) image size.

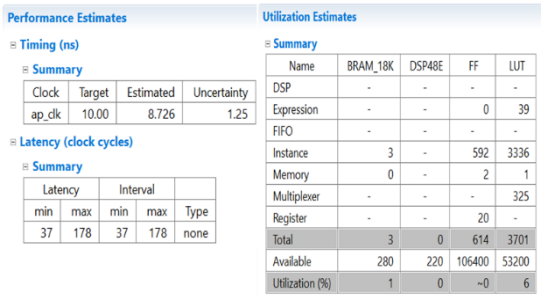


Figure 12: Run synthesis\ apply BF-decoder using loop unrolling directive (16*16) image size

Performance and utilization results of BF-Decoder in the Vivado_HLS program for image (16*16) without optimization is illustrated

in Figure 11. The synthesis results of the decoder image (16*16) using directives are shown in Figure 12.

B-Synthesis results decoder in Vivado_HLS program for image (16*16)

The comparisons between different resources of a decoder of the (16*16) image when using optimization and without using it, are illustrated in Figure 13.

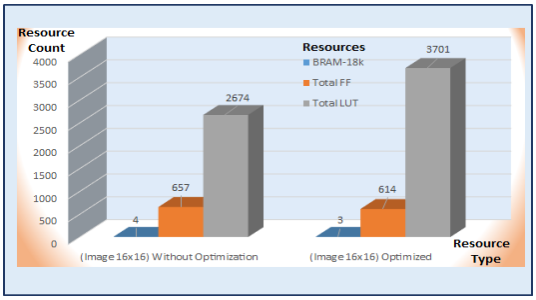


Figure 13: Resources of Decoder of Image (16x16)

The equation below has been used as a formula to determine the decoder's throughput.

$$\text{Thrughput} = \frac{\text{code rate} * \text{Fmax}}{\text{No. of iteration} * \theta} \qquad (7)$$

Equation 7 elements: (Fmax = 1 / Estimated Time) is the decoder's highest operating frequency as determined by the FPGA implementation. The number of clock cycles needed to finish an iteration is represented by θ. Code rate R = (n – m) / n. The matrix's elements can be either 1 or 0, and the code rate that was used was 0.5. Programs typically use an H matrix with a (m, n) (8,16) in size. There were 16 bits in the code. So, the throughput of the BF-Decoder algorithm was computed into two situations. It was 5.15 Mbps when the optimization technique (loop unrolling) was applied and a 16*16 image, compared to 1.29 Mbps when not. The Throughput has increased, as we can see.

The synthesis report includes details on performance estimates, critical paths, and resource utilization (FPGA slices, LUTs, DSPs, BRAMs, etc.). The BF-Decoder design utilization report is displayed in Table 3.

5. CONCLUSION

This paper addresses the critical challenges of reliable image transmission in modern communication systems, emphasizing the importance of Low-Density Parity-Check (LDPC) codes, a prominent type of channel coding that reflects their exceptional error-correcting capabilities, positioning them as a key element in Forward Error Correction (FEC) codes.

Table 3: The design report of the utilization of the BF-Decoder

Image Size	16 * 16	
Resources	Without Optimization	Optimized
Target Time - Uncertainty	8.75 ns	8.75 ns
Estimated Time	7.196 ns	8.726 ns
Min latency	362	37
Max latency	861	178
Min Latency in Instance	311	19
Max latency Instance	810	160
BRAM-18k	4	3
Total FF	657	614
Total LUT	2674	3701
loop	2	N/A
LUT in Multiplexer	164	325
Total bits in Multiplexer	82	171
FF in Register	41	20

Also, it successfully proposes and implements image transmission through LDPC architectures for both the channel encoder and decoder, utilizing the Effective Encoding of LDPC Codes algorithm for encoding and the bit flipping LDPC algorithm for decoding. The powerful and modern Vivado HLS with High-Level Synthesis (HLS) and the C programming language form the foundation of the design process, allowing for efficient code optimization through HLS techniques such as loop unrolling. The outcomes of the study reveal that the incorporation of optimization techniques, specifically loop unrolling, leads to increased throughput while simultaneously reducing resource requirements.

The throughput of the BF algorithm is 5.15 Mbps when the Loop Unrolling Directive is used, and 1.29 Mbps when it is not.

One of the important issues that must be taken into consideration is to put the image in an integer matrix to facilitate the following steps of the process. Then, after decoding, the image will be recovered to the structured form.

REFERENCES

[1] I. Develi and Y. Kabalci, "A comparative simulation study on the performance of LDPC coded communication systems over Weibull fading channels," Journal of Applied Research and Technology, vol. 14, pp. 101-107, 2016, doi: 10.1016/j.jart.2016.04.001.

[2] M. Yasoubi, “An efficient hardware implementation of LDPC Decoder,” M.S. dissertation, Electrical and Computer Engineering Dept. Concordia University, Montréal, Québec, Canada, 2020.

[3] Arwa H. Ashou, and Dhafir A. Alneema, "FPGA Hardware Design of Different LDPC Applications: Survey," Asian Journal of Computer Science Engineering, vol. 6, pp. 35-44, 2021.

[4] M. G. Prasad, C. C. Reddy, and J. C. Babu, "VLSI Implementation of decoding algorithms using EG-LDPC Codes," Procedia computer science, vol. 115, pp. 143-150, 2017, doi: 10.1016 /j.procs.2017.09.119.

[5] P. Raju and P. S. Prasad, "Design of an LDPC Decoder and Its Performance," International Journal of Electrical and Electronics Communication, vol. 1, no. 1, 2017.

[6] N. J. Gaurihar, I. R. Khadse, T. S. Ghonade, A. Borkar, A. Singh, and M. Patil, "Design and implementation of LDPC codes and turbo codes using FPGA," Int. Res. J. Eng. Technol., vol. 3, pp. 1683-1687, 2016.

[7] Marwan Abdulkhaleq Al-yoonus, Saad Ahmed Al-Kazzaz, "FPGA-SoC Based Object Tracking Algorithms: A Literature Review," Al-Rafidain Engineering Journal (AREJ), Volume 28, Issue 2, Page 284-295, September 2023, doi: 10.33899/rengj.2023.138936.1243.

[8] B. M. K. Younis, B. Sh. Mahmood, and F. H. Ali, "Reconfigurable Self-Organizing Neural Network Design and its FPGA Implementation," Al-Rafidain Engineering Journal (AREJ), vol. 17, no. 3, pp. 99–115, Jun. 2009, doi: 10.33899/ rengj.2009.42925.

[9] R. Marzotto, P. Zoratti, D. Bagni, A. Colombari, and V. Murino, "A real-time versatile roadway path extraction and tracking on an FPGA platform," Computer Vision and Image Understanding, vol. 114, no. 11, pp. 1164–1179, Nov. 2010, doi: 10.1016/j.cviu.2010.03.015.

[10] G. Conti, M. Quintana, P. Malagón, and D. Jiménez, "An FPGA Based Tracking Implementation for Parkinson's Patients," Sensors, vol. 20, no. 11, p. 3189, Jun. 2020, doi: 10.3390/s20113189.

[11] M. Amiri, F. M. Siddiqui, C. Kelly, R. Woods, K. Rafferty, and B. Bardak, "FPGA-Based Soft-Core Processors for Image Processing Applications," Journal of Signal Processing Systems, vol. 87, no. 1, pp. 139–156, Apr. 2017, doi: 10.1007/s11265-016-1185-7.

[12] Zahraa T. Al-Mokhtar; Farah N. Ibraheem and Hassan F. Al-Layla, "A Review of Digital Image Fusion and its Application, " Al-Rafidain Engineering Journal (AREJ), Volume 26, Issue 2, Page 309-322, October 2021, doi: 10.33899/rengj.2021.127928.1055.

[13] [1] A. V. Bukit and Wirawan, "3D video coding development based on FPGA platform Xilinx Zynq-7000," 2017 International Seminar on Intelligent Technology and Its Applications (ISITIA), Aug. 2017. doi:10.1109/isitia.2017.8124050.

[14] Gururaj, Bharathi, and G. N. Sadashivappa. "Channel encoding system for transmitting image over wireless network," International Journal of Electrical and Computer Engineering, vol. 10, pp. 4655, 2020, doi: 10.11591/ijece.v10i5.pp4655-4662.

[15] Nadzri, Muhamad Muzakkir Mohd, and Afandi Ahmad. "SoC FPGA-Based Rapid Prototyping of Compressed, Secured and Wireless Image Transmission for Wildlife Surveillance System," International Journal of Integrated Engineering, vol. 14, pp. 276-285, 2022.

[16] Amer T. Ali and Dhafir A. Alneema, "Design Analysis of Turbo Decoder Based on One MAP Decoder Using High Level Synthesis Tool," Al-Rafidain Engineering Journal (AREJ), vol. 25, pp. 70-77, 2020, doi: 10.33899/rengj.2020.126801.1022.

[17] N. J. Gaurihar, I. R. Khadse, T. S. Ghonade, A. Borkar, A. Singh, and M. Patil, "Design and implementation of LDPC codes and turbo codes using FPGA," Int. Res. J. Eng. Technol., vol. 3, Issue: 03 , pp. 1683-1687, 2016.

[18] B. S. Reddy and V. S. R. Rao, "FPGA Implementation of LDPC Encoder and Decoder using Bit Flipping Algorithm," International Journal of Science and Research, vol. 6, Issue: 9, ,pp. 1683-1690, 2017.

[19] Murillo, Raul, et al. "Generating Posit-Based Accelerators With High-Level Synthesis," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 70, Issue 10, pp. 4040-4052, October 2023, doi: 10.1109/TCSI.2023.3299009.

[20] H. Malin, "High Level Synthesis for ASIC and FPGA," Master's Thesis, Electrical and Information Technology Dept., Lund University, EITM01 20211, 2023.

[21] E. N. de Souza and G. L. Nazar, "Cost-effective Resilient FPGA-based LDPC Decoder Architecture," IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 84-89, 01-03 July 2019, doi: 10.1109/IOLTS.2019.8854457.

نقل الصور من خلال معمارية ترميز القناة

زينب علي الخياط zainab.21enp6@student.uomosul.edu.iq

ظافر عبد الفتاح dhafir.abdulfattah@uomosul.edu.iq

قسم هندسة الحاسوب، كلية الهندسة، جامعة الموصل، الموصل ، العراق

تاريخ الاستلام: 14 يناير 2024 استلم بصيغته المنقحة: 13 فبراير 2024 تاريخ القبول: 4 مارس 2024

الملخص

يحتاج نقل الصور في أنظمة الاتصالات الحديثة إلى تشفير أخطاء سريع ومنخفض وآليات نقل حتمية .بالنسبة للمهندسين، يعد الاتصال الموثوق به عبر قناة صاخبة مشكلة طويلة الأمد ولكنها صعبة .أحد الأنواع المهمة لتشفير القنوات هو رموز فحص التكافؤ منخفض الكثافة (LDPC) *التي تأخذ في الاعتبار رموز الكتلة الخطية* .(LBC) *نظرًا لقدرتها الفائقة على تصحيح الأخطاء، تعد رموز* LDPC *من بين أكواد تصحيح الأخطاء الأمامية* (FEC) *الأكثر استخدامًا على نطاق واسع .الغرض من هذا البحث هو إنشاء معمارية* LDPC *لتشفير قناة الإرسال، ووحدة فك ترميز القناة المقابلة في جهاز الاستقبال لنقل الصور . تدمج الدراسة خوارزمية التشفير الفعال لرموز* LDPC *لجهاز التشفير وخوارزمية* Bit Flipping Codes LDPC *لجهاز فك التشفير،* Vivado HLS *هي الأداة المستخدمة في هذا العمل، وتقنية* HLS Loop Unrolling *الأمثل المستخدمة لإعطاء تعليمات المركب حول كيفية التنفيذ .قسم معين من التعليمات البرمجية، يمكن للمصمم تحسين التطبيق بسرعة وسهولة، ونتيجة لذلك، يتم التحسين مباشرة على التعليمات البرمجية المصدر .بالإضافة إلى ذلك، يقترح تطبيق تقنيات التحسين مثل فتح الحلقة على كل تصميم .يتم استخدام لغة البرمجة* C *والتوليف عالي المستوى* (HLS) *لإنشاء جميع البنى.*

الكلمات الدالة :

نقل الصور؛ التشفير الفعال لـ LDPC؛ بت التقليب LDPC؛ برنامج Vivado HLS.