Kufa Journal of Engineering
Vol. 13, No. 1, January 2022, P.P. 82-101
Article history: Received 16 October 2021, accepted 28 November 2021

REDUCING LDPC DECODER COMPLEXITY BY USING FPGA BASED ON MIN SUM ALGORITHM

Butheena R. kadhim[1] and Fatih Korkmaz[2]

[1] MSc. student, Department of Electrical - Electronics Engineering, Çankırı Karatekin University, Turkey. Email: butheenarheem88@gmail.com

[2] Asst. Prof. Department of Electrical - Electronics Engineering, Çankırı Karatekin University, Turkey. Email: fkorkmaz@karatekin.edu.tr

HTTPS://DOI.ORG/10.30572/2018/KJE/130105

ABSTRACT

For double fields and large code lengths, Low-Density Parity-Check (LDPC) code approaches Shannon–limit execution. The goal of this project is to present an LDPC calculation for Min Sum (MS) decoding and equipment execution inquiry within a communication framework that has been proposed. The MS calculation principally utilizes the base and expansion finding procedure. The quantity of increases is subsequently altogether diminished, which tends to decrease the execution intricacy. The consequences of the reenactment show that the proposed MS interpreting calculation performs the same as the translating of the Sum-Product Algorithm (SPA) while keeping up with the principle highlights of the MS disentangling. The further developing execution by diminishing the number of stages in the deciphering system, diminishing the intricacy of the implementation of the Field Programmable Gate Array (FPGA).

KEYWORDS: LDPC, MS, SPA, Log Domain.

Article information: This manuscript was handled by Abbas Al-Hdabi, Managing Editor, with the assistance of Esam Alkaldy Associate Editor.

1. INTRODUCTION

LDPC codes are a type of error-correction code that Gallagher initially proposed in 1962 during his research (Shannon, 2001). However, because advanced computing methods were not available, functional implementation was not possible. Mackay and Neal re-invented LDPC codes (MacKay and Radford, 1996; MacKay, 1999) using increases in processing capacity and the formulation of new theories. LDPC codes were linear block codes that utilised sparse parity-check matrices. Both the rational distance properties and the relatively low decoding algorithm complexity are induced by the parity check matrix's low-density structure (Tomlinson et al., 2017). LDPC codes have exceptional efficiency, flexibility, and parallel capabilities, resulting in superior hardware implementation. An LDPC code may be decoded with a high degree of parallelism, making it ideal for high data rate applications like wide-band wireless multimedia communications and all modern communications applications (Kalsi et al., 2012).

Gallagher presented the SPA as the basic decoding technique in 1962. When it comes to real-time implementations like FPGAs (Roh et al., 2016; Ceroici, and Vincent, 2014), the SPA in (Mooij, Joris M., and Hilbert, 2005; Chen, et al., 2004) provide further addition and multiplication steps that render the computation complex.

As a consequence, to promote the functional implementation of codes, a simplified version of the SPA was introduced. MS was an algorithm which is designed to reduce the complexity of hardware (Angarita, et al., 2014). Approximation is done at check nodes for sophisticated computations by comparison and summation operations in this procedure. N. Weiberg, (2014), has spent a lot of time working on the MS method and has demonstrated that the implementation complexity is greatly decreased. In order to enhance its performance with respect to SPA (Ahmed and Elsabrouty, 2014; Hatami, Homayoon, et al., 2020; Jianguang et al., 2005), several updated versions of MS decoding algorithms were later proposed. Due to its low complexity, the MS decoding approach is commonly used in many digital broadcasting applications. In this case, a scaling factor is being used in the case of optimized MS (Jun, and Chugg, 2005) to reduce the error induced by the minimum operation and thereby improve the accuracy.

The proposed system's performance and hardware implementation complexity are examined in this study with the MS Algorithm as decoder techniques which has better performance from other SP algorithms and low complexity. Using Xilinx System Generator (SG) packages, the practical viability of the soft decision MS LDPC code method in the Additive

White Gaussian Noise (AWGN) channel is verified, which offers a number of advantages over the previous implementation platforms on the Kentix 7 FPGA kit.

2. A PROPOSED COMMUNICATION SYSTEM

In this research, a proposed communication system included an LDPC encoder on the transmitter side, while the decoding on the receiver side. The decoder is constructed from Log Domain and MS decoders. To clear out the idea, it will be explained in the following part of the system model shown in Fig. 1.

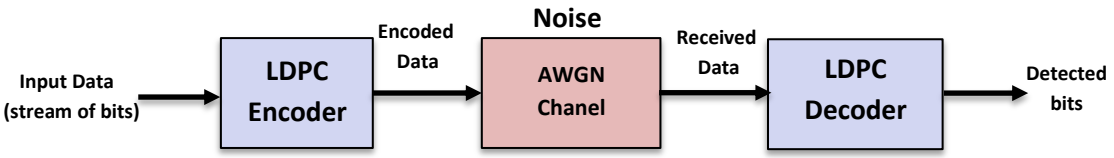

Fig. 1. LDPC system model.

The source produces a bits stream which will be transmitted and subjected to the noise. In order to correct the errors which are stem from such noise, LDPC will be encoded such bits stream as follow:

2.1. LDPC Encoder

The binary LDPC code is a sparse H parity-check matrix M x N that defines a linear block code. The rows of the parity check matrix represented check nodes, whereas the columns represented variable nodes. Each check node M corresponds to one bit of the codeword, and each variable node N corresponds to one parity check equation. Johnson, (2006); Mosleh, (2011) Graph edges were utilized to connect variable nodes and represent non-zero items in the H matrix in Johnson, (2006) and Mosleh, (2011). H contains a modest number of nonzero entries; in general, it is linear in block length n, and regular and irregular types of parity check matrices can be utilized. We'll be using irregular codes in this course. For irregular codes, the equivalent H matrix has dc ones in each row and dv ones in each column. This means that each codeword bit participates actively in specific dc parity check equations, and that dv codeword bits are employed concisely in each of these check equations. This is an example of an irregular parity check matrix with some of its corresponding parity check equations with parity-check matrix is($dc = 10$ and $dv = 20$).

$$H = \begin{array}{c} \begin{matrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 & X_9 & X_{10} \end{matrix} \\ \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&1&1&1&1&0 \\ 0&0&0&1&0&0&0&0&0&0&1&1&0&1&0&0&0&0&1&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&1&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&0 \end{bmatrix} \end{array} \quad (1)$$

$$\boldsymbol{c_1} = \boldsymbol{s_3} + \boldsymbol{s_6} + \boldsymbol{s_9} \quad (2)$$

$$\boldsymbol{c_2} = \boldsymbol{s_5} + \boldsymbol{s_6} + \boldsymbol{s_7} + \boldsymbol{s_8} \quad (3)$$

$$\boldsymbol{c_3} = \boldsymbol{s_1} + \boldsymbol{s_2} + \boldsymbol{s_6} + \boldsymbol{s_7} + \boldsymbol{s_8} + \boldsymbol{s_9} \quad (4)$$

$$\boldsymbol{c_4} = \boldsymbol{s_1} + \boldsymbol{s_2} + \boldsymbol{s_4} + \boldsymbol{s_9} + \boldsymbol{s_{10}} \quad (5)$$

$$\boldsymbol{c_5} = \boldsymbol{s_1} + \boldsymbol{s_3} + \boldsymbol{s_6} \quad (6)$$

where $\boldsymbol{c_1} \ldots . \boldsymbol{c_{10}}$ represent the parity check bits equation.

The Tanner graph represents a graph related to a parity check matrix and includes two kinds of nodes: *M* nodes (which represent the N bits of a codeword) and N nodes (which represent the N bits of a codeword) (representing the parity constraints). As a result, the parity check matrix is represented graphically. Fig. 2 depicts the bipartite graph for the parity check matrix (6x12) (Johnson, 2006; Mosleh et al., 2020).

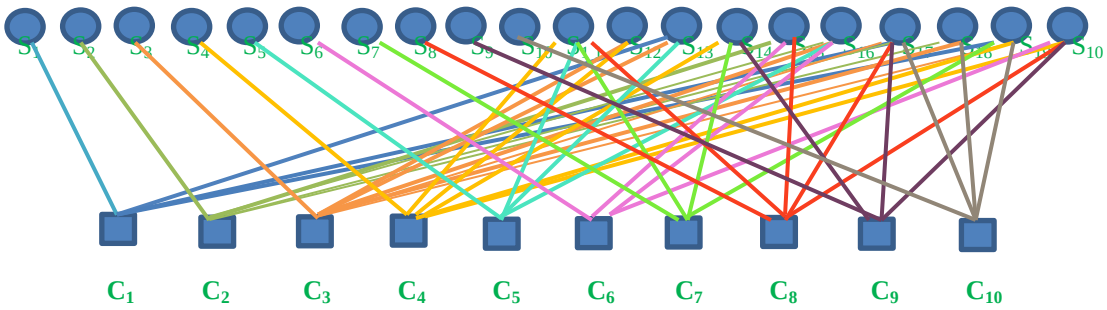

Fig. 2. An irregular parity check matrix is represented by a bipartite graph.

Let's *C* have been an irregular LDPC code with length *N* and dimension *K* where parity-check matrix *H* contains exactly dv 1's in each column (column weight) with M=N-K rows and N columns as well as exactly dc 1's in each row (row weight). $H_{m,n}$ is the m_{th} row value and.n_{th} column in H. The set of bits that are involved in the check is denoted: $N_m = \{n: H_{m,n} = 1\ \}$. The set of checks that participate in the bits $M_m = \{m: H_{m,n} = 1\ \}$. Assuming the code word, $c = [c_1, c_2, c_3, \ldots\ldots, c_N]^T$.

2.2. AWGN Channel

The codeword is mapped using Binary Phase Shift Keying (BFSK) modulation to such a signal constellation before transmission to obtain the vector, $t = [t_1, t_2, t_3, \ldots\ldots, t_N]^T$, which would be transmitted via an AWGN channel with variance (Mosleh, 2011).

$$\sigma^2 = N_0/2, \quad (7)$$

$$r = [r_1, r_2, r_3, \ldots\ldots, r_N]^T \quad (8)$$

Where

$$r_n = t_n + v_N \quad (9)$$

Here, the AWGN of zero means is v_N.

2.3. LDPC Decoder

To grasp the concept of the decoder employed in this research, we must first explore the SPA's necessary context theory before presenting the MS decoded technique on the receiver side. All SPA belief propagation algorithms function by processing received symbols in substring phases, which may be thought of as a horizontal step along the Tanner graph followed by a vertical step to increase the decoded code symbol's dependability. At the end of each decoding cycle, the estimated reliability measurements of the code symbols are used as inputs for the next iteration. The decoding iteration method continues once a specified halting criteria is reached (Mosleh et al., 2020). A brief description of the log domain and MS algorithms is provided in the following subsections.

A. Log Domain Sum-Product Decoding Algorithm

The Log Domain decoding algorithm's mathematical model is as follows (Richardson and Rudiger, 2003).

Step 1: **Initialization:** Let Lci_n is the a priori information of the nth symbol which represents the Log-Likelihood Ratio (LLR) from an AWGN channel. It initially assigned for the computation to the variable node unit $alpha_{m,n}$ and $beta_{m,n}$ will determine the sign value for eachLci_n, whether it is positive, negative, or equal to zero, as well as the absolute value.

$$Lci_n = \frac{-4r_x}{N_0} \quad (10)$$

$$alpha_{m,n} = sign\ (Lci_n) \quad (11)$$

$$beta_{m,n} = |Lci_n| \quad (12)$$

Step 2: **Check node update (Horizontal Step):** Let $Pi_{n,m}$ denote the probability of achieving parity-check m in the case when bit n is expected to be a 1 for the LLR.

$$Pi_{m,n} = \log(\frac{e^{betaij_{m,n}} + 1}{e^{betaij_{m,n}} - 1}) \quad (13)$$

Compute the summation of $Pi_{m,n}$ excluding the bit n.

$$SP_{m,n} = \sum_{\mathrm{n}\in B_m, \mathrm{n}'\neq \mathrm{n}} Pi_{m,n'} \quad (14)$$

Then compute $PiS_{m,n}$:

$$PiS_{m,n} = \log(\frac{e^{SP_{m,n}} + 1}{e^{SPi_{m,n}} - 1}) \quad (15)$$

Get products of $alphaij_{n,m}$ excluding the bit n:

$$Pr_{m,n} = \prod_{\mathrm{n}\in B_m, \mathrm{n}'\neq \mathrm{n}} alphaij_{m,n}{}' \quad (16)$$

Finally, compute $Fji_{m,n}$

$$Fji_{m,n} = PiS_{m,n} \times Pr_{m,n} \quad (17)$$

Step 3: **Variable node update (Vertical Step):** The messages $Fji_{m,n}$ from the check node update unit will be transferred to the variable node update unit for the estimation of a codeword based on the number of 1s in all rows of the *H* matrix.

$$L_{Qi} = Lci_n + \sum_{m\in A_n} Fji_{m,n} \quad (18)$$

For the n^{th} digit, L_{Qi} represent the total log-likelihood ratio.

Step 4: **Tentative decoding:** With the messages received from the check node update unit, the code word is tentatively decoded for each symbol as

$$vhat = \begin{cases} 1, & L_{Qi} < 0 \\ 0, & L_{Qi} \geq 0. \end{cases} \quad (19)$$

Step 5: **Decision:** The tentatively decoded symbols are cross verified with the parity check matrix, to validate the decoded codeword using

$$CH^T = 0$$

B. Min Sum Decoding Algorithm

The MS Algorithm is a modified version of this SPA, with the action of the check node simplified to lower the algorithm's complexity significantly. The extrinsic messages (between variable and check nodes) and the quantized intrinsic message (also known as the LLR) are

both the same length in MSA. As a result, the hardware implementation complexity, notably the decoding node interconnects for MSA, grows as the quantized message length grows. Equations 20, 21, 22, and 23 show the actions of the initialization and check nodes, respectively (Kenneth et al., 2007).

MSA initialization Process:

$$Lci_n = -r_x \tag{20}$$

MSA check node operation:

$$Ai_{m,n} = min_{n\in B_m, n\prime\neq n}(betaij_{m,n\prime}) \tag{21}$$

$$Pr_{m,n} = \prod_{n\in B_m, n\prime\neq n} alphaij_{m,n}{}' \tag{22}$$

$$Fji_{m,n} = A_{m,n} \times Pr_{m,n} \tag{23}$$

3. HARDWARE IMPLEMENTATION OF PROPOSED SYSTEM DESIGN

The LDPC system model with the MS decoder communication system is shown in Fig. 3. Xilinx SG is used to build each block in the system, with a master clock duration of 10 ns. The suggested system had three major components: a transmitter, a channel, and a receiver. The Bernoulli Binary Signal Generator can be used to build a random bit generator with a code rate of 0.5 on the transmitter side. The LDPC block will encode this binary data by solving the parity check equation for each row of the H matrix using the systematic form of the H matrix. To create a modulated signal, the resulting codeword will be passed to BFSK. AWGN noise, which is often used in experimental applications due to its simplicity, will alter the modulated signal.

To recover the bit binary stream on the receiver side, the LDPC decoder with MS decoded technique was used. The following subsections detail the general system implementation, which will be explained in depth in the following subsections.

3.1. Transmitter Section

In this section, Bernoulli Binary Signal Generation, serial to parallel, encoder block, and mapping block will be used. Every block is designed by means of Xilinx SG block in MATLAB/Simulink environment.

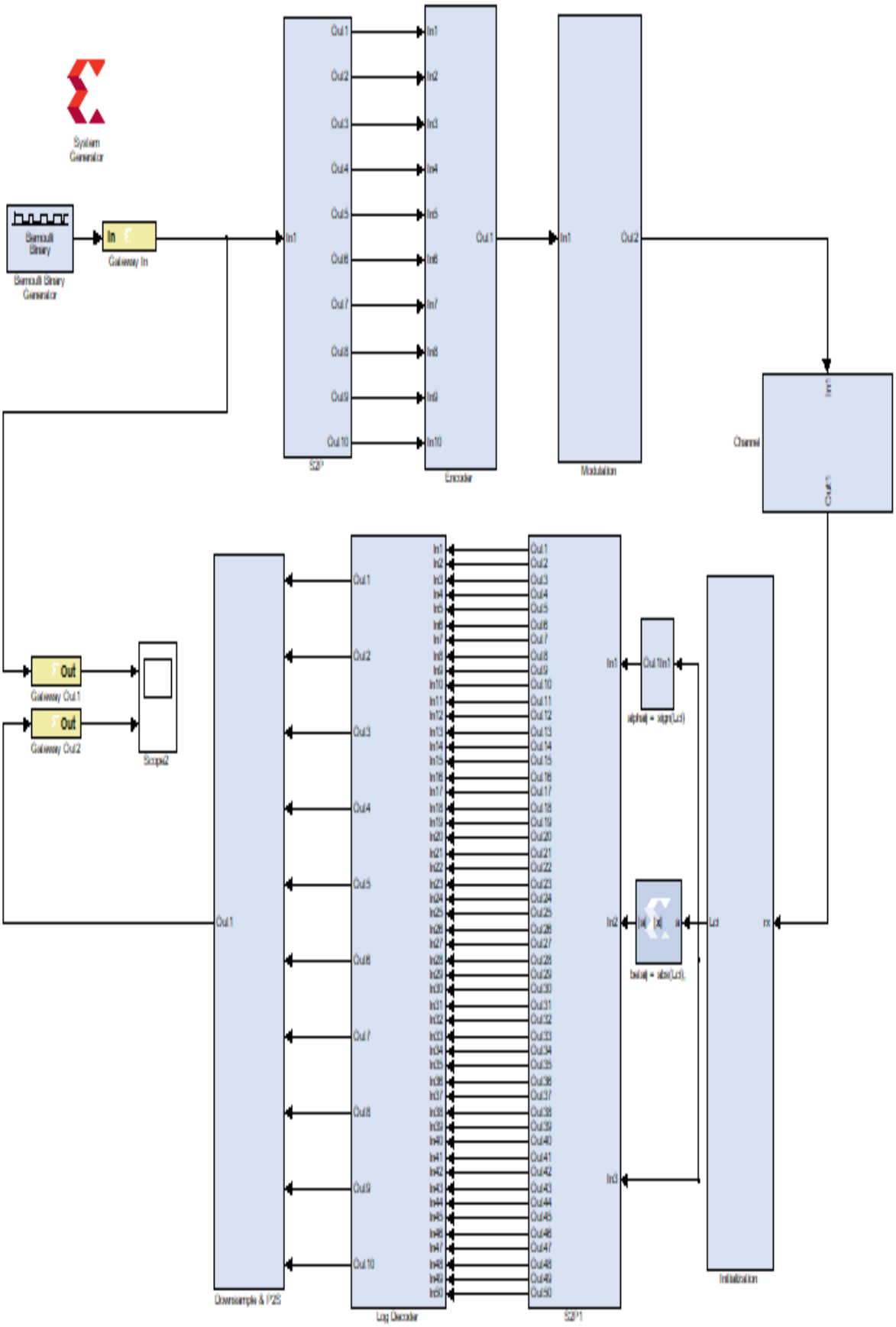

Fig. 3. The Xilinx SG corresponds to the proposed block diagram of Fig. 1.

a. Bernoulli Binary Signal Generation

The Bernoulli Binary Signal with a length of k=10 was generated using the Simulink block Bernoulli Random Binary Number Generator. The Sample time of this block is equal to 1, the initial seed value is 61 with a 0.5 probability of zero. The output data type is Boolean, and it is passed through a gateway block to the Serial to Parallel block.

b. Serial to parallel converter Block

The serial to parallel block, as shown in Figure 4, turns a series of serially presented data into single samples at the output. The slice block was used to extract a given bits range of each input sample from the Serial to Parallel output, which is a 10-bit symbol.

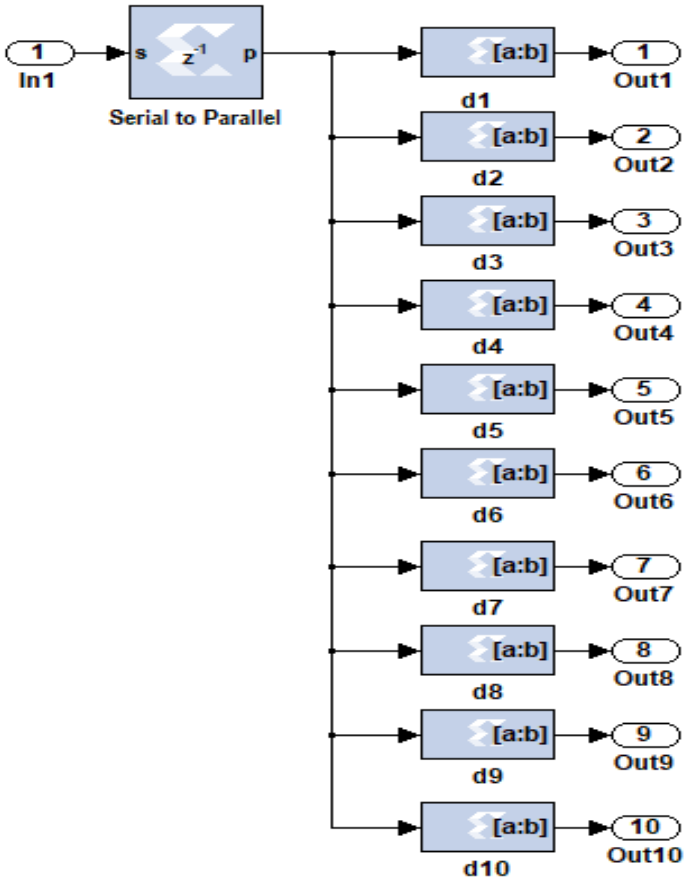

Fig. 4. Serial to Parallel converter Xilinx SG block.

c. Encoder Block

Fig. 1 depicts the hardware implementation of such a block. 5 which are used in accordance with the Eqs. Two, three, four, five, and so on use ten XOR gates to generate ten parity check bits, depending on the input message. The concat block takes a 20-bit code word and concatenates two or more input bits to create a symbol in the output. Finally, a parallel to serial block serves as the code word's output block, converting each sample supplied in the input to numerous samples delivered serially in the output.

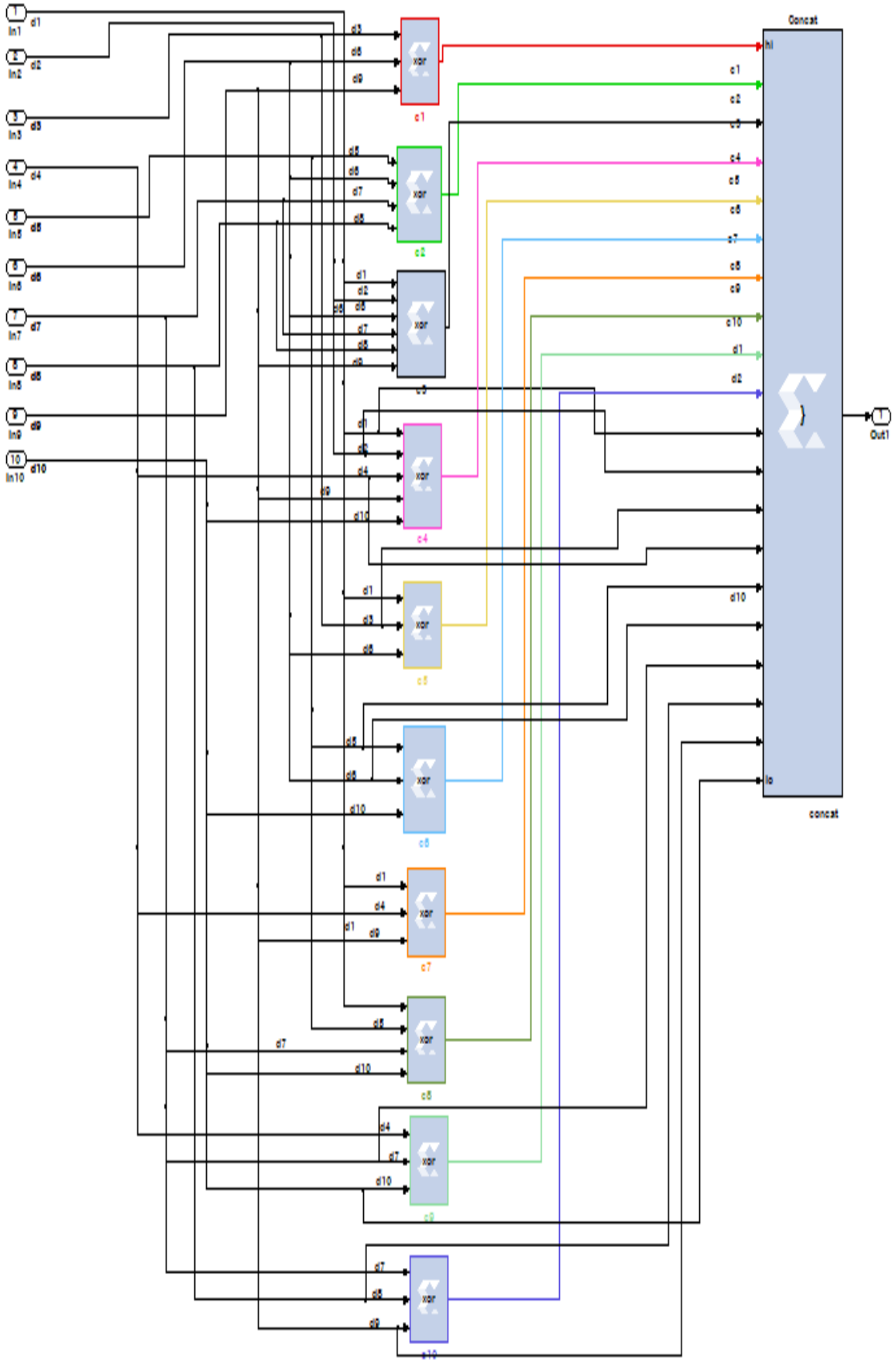

Fig. 5. The Xilinx SG block diagram of Encoder Block.

d. Mapping Block

Each parallel bit is mapped using mapping blocks. Each parallel data bit must have served as a ROM block address, indicating either the 0 or 1 input corresponding to the ROM's initial value vector [-1 1]. The data type of the output is signed. The pin on the ROM block was activated with a delay to mask all serial bits until they were ready to be mapped. The Xilinx SG block diagram for the mapping function is shown in Fig. 6.

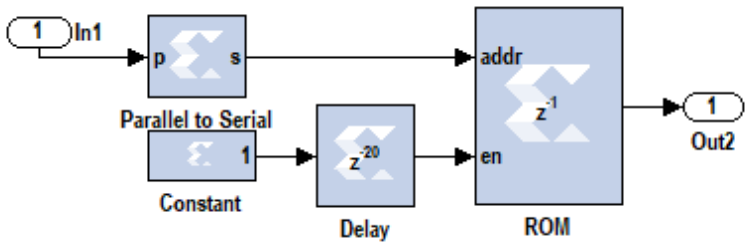

Fig. 6. Xilinx SG block diagram of Mapping Block.

3.2. AWGN Channel

When a modulated signal is broadcast across a channel, it will be degraded by noise in the channel represented by AWGN, n k. The noise at question, according to Johnson (2006), is statistically random radio noise with a wide frequency range. The seed value for this block is equal to 512. The hardware-implemented of such channel according to equation 6. is illustrated in Fig. 7.

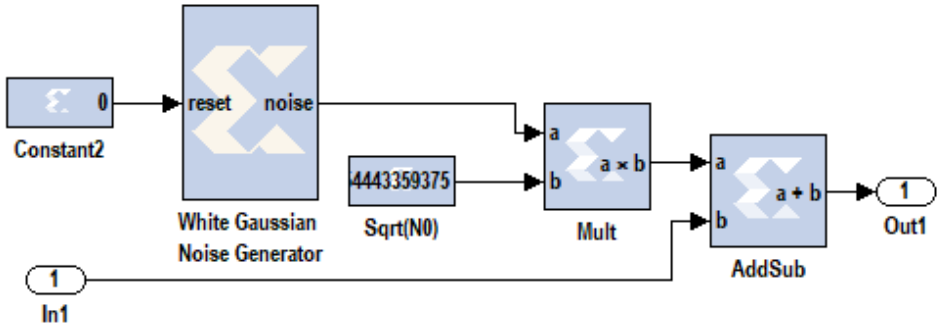

Fig. 7. The Xilinx SG block diagram of AWAGN Channel.

3.3. Receiver Section

In this section, using Xilinx SG block sets, the initialization block, serial to parallel converter, MS Decoder Block, and Down sample were introduced and implemented.

a. Initialization Block

Based on the Equations, compute the absolute value and sign value., this block is implemented using the Mult, Mux, and Relational Xilinx SG blocks. The numbers 20, 11, and

12 are the ones to remember. Figs. 8 and 9 depict the initialization block and signature block processes, respectively.

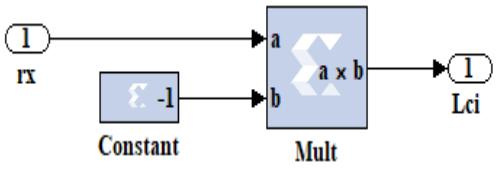

Fig. 8. Initialization process (Lic).

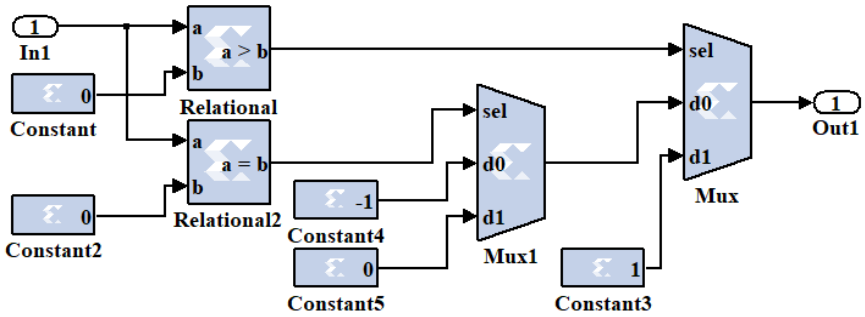

Fig. 9. Sign block details.

b. Serial to Parallel Converter

A serial to parallel converter that uses 20 shift registers and their related delay blocks, enable blocks, and LFSR Xilinx blocks converts the serial sequence in this block to parallel samples. The 20 samples were latched to the 20 shift registries by triggering the enable pin. In this case, the delay is employed for synchronization. To store the entire sample, the LFSR was utilized to create a pattern. The Xilinx SG S2P block is seen in detail in Fig. 10.

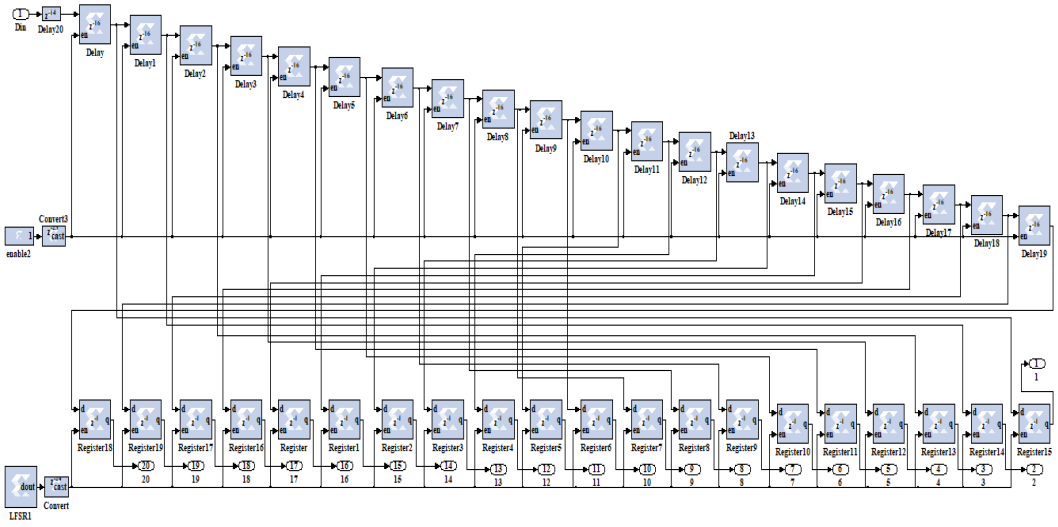

Fig. 10. The Xilinx SG block diagram of 20-sample Serial to Parallel Block.

c. Min Sum Decoder Block

The vertical and horizontal step techniques are implemented using Xilinx SG (Multi, Add-Sub, Mux, and Relational) blocks to manage the value of the variable node and check node, according to Eqs.18, 21, 22, and 23 as illustrated in Fig. 11 and 12 by taking the design for the first row and the twelve columns as an example for the vertical and horizontal step Xilinx SG block implementation. In Figs. 12-16, the specifics for each of the blocks in Fig. 11 can be declared.

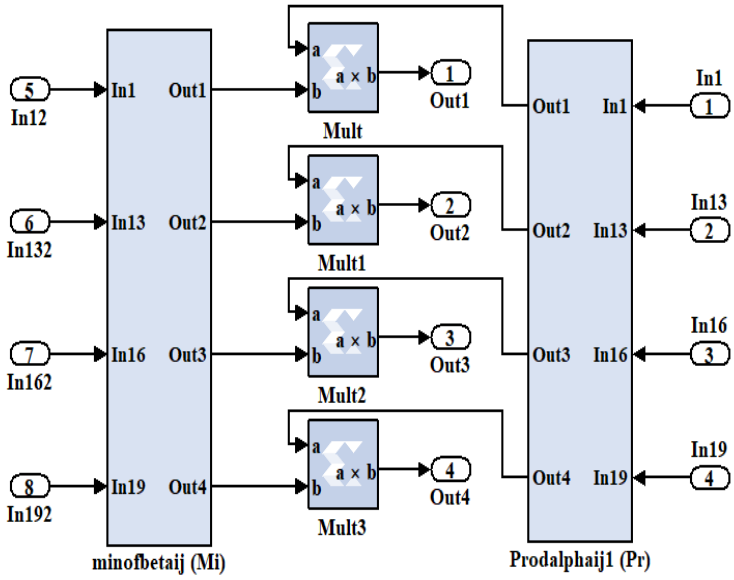

Fig. 11. The first row's horizontal step process.

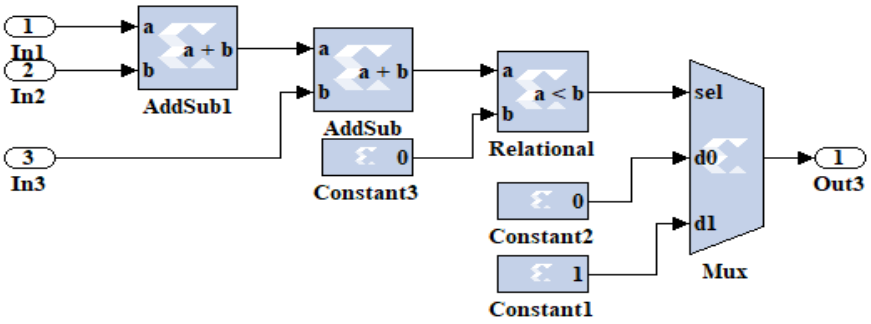

Fig. 12. Column 12's vertical step blocks.

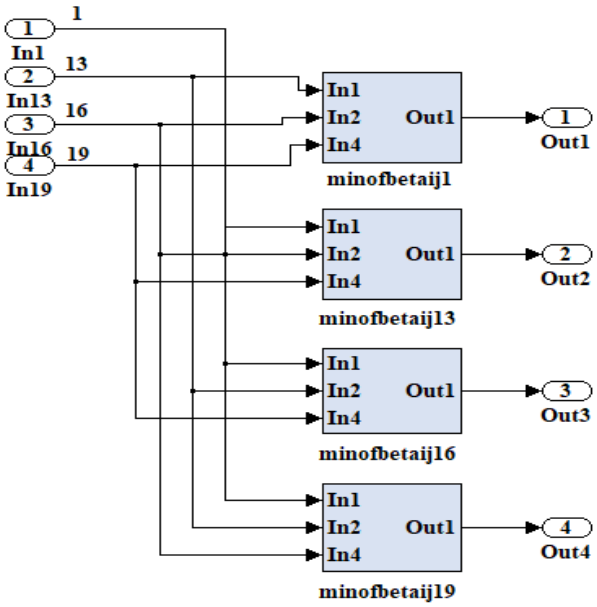

Fig. 13. Details of Mi block.

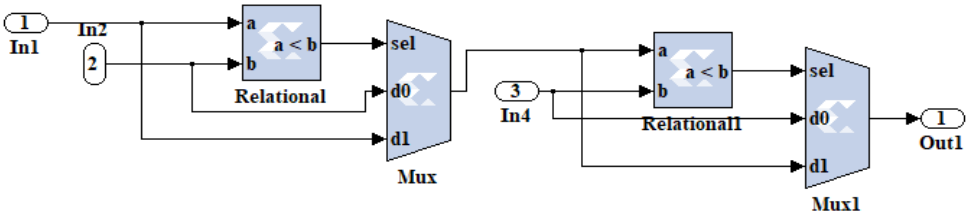

Fig. 14. Blocks Mi 1,13,16, and 19 in detail.

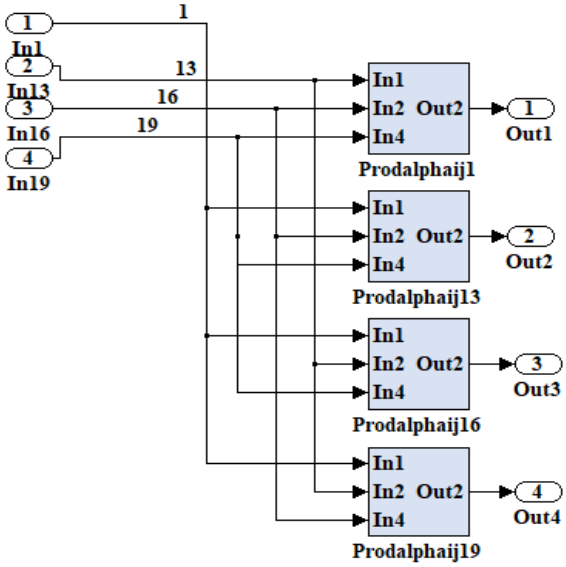

Fig. 15. Pr block's details

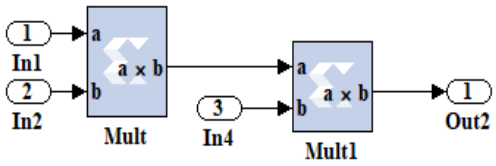

Fig. 16. Pr 1,13,16, and 19 blocks described detail.

d. Down sample

The output bits of the LDPC decoder architecture were connected to a down-sample block composed of Xilinx SG blocks with a sampling rate of 20 to control the rate of the output signal at the receiver. After passing through concert and Parallel to Serial blocks to obtain the information data shown in Fig. 17, these bits from the output of the down-sample blocks will be changed from parallel bits to stream bits.

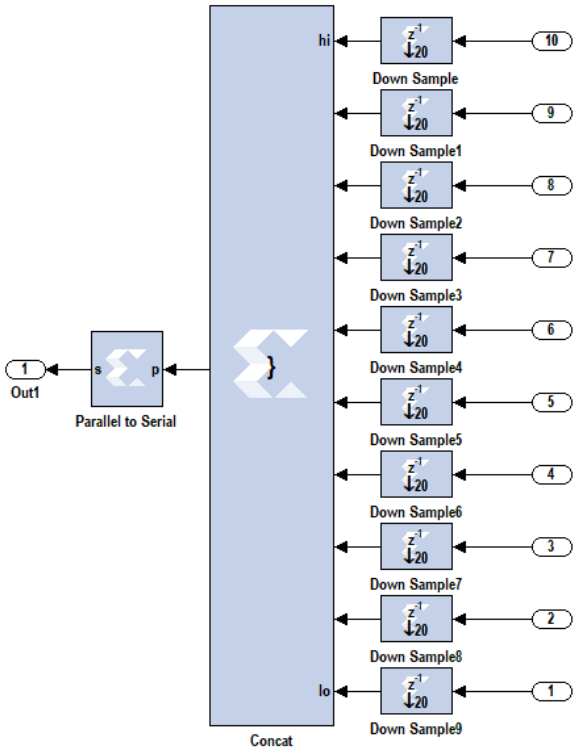

Fig. 17. The Xilinx SG block diagram of Down Sample Block.

4. Software Simulation Results

Bit Error Rate (BER) as a function of Signal-to-Noise Ratio (SNR) was used to evaluate the performance of the proposed system's codes. The irregular (256,512) LDPC codes with code rate 1/2 are analyzed. Codes were transmitted after BPSK modulation on the AWGN channel. To terminate the simulation with the optimum parameter, the maximum number of iteration ($iter$) set to 6, the No. of one in each column (w_c) set to 3 and the frame length (F) set to

100. The results of the simulation show that the efficient MS algorithm achieves better performance than the log domain SPA. Fig.18 clear out that for the BER value, 10^{-3}, a comparison between MS algorithms can achieve 0.5 dB decoding gain over the log domain algorithm. The simulations were performed using the MATLAB Ver. 2019a platform. As a function of SNR, BER evaluated the performance of codes obtained.

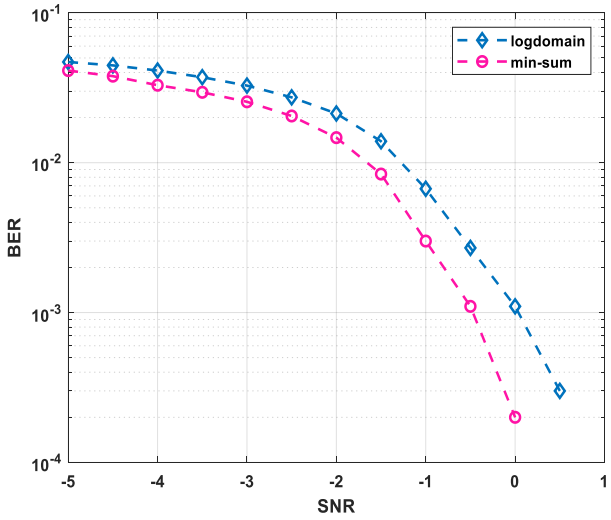

Fig. 18. Bit Error Rate performance of LDPC code (256, 512) for MS and Log Domain.

5. FPGA Synthesis Results

By selecting a Kintex 7 device, the bit stream file or VHDL code file was generated. Using the ISE 14.5 program and MATLAB Ver 2912a, the devices used to construct this system are illustrated. The minimum period is 25.088 ns, the maximum frequency is 39.860 MHz with throughput equal to 797.2Mbs. The hardware resources are required for MS decoder techniques are listed in Table 1.

To compare the whole system's degree of complexity, Table 2 summarized the resource device consumption in the (MS and Log Domain) soft-decision decoder. The MS Decoder uses fewer Look Up Tables (LUTs) from the comparisons because it has no additional variables to store. This decoder's operating frequency is less than other techniques. Due to their BER performance and ideal resource usage, it is easy to conclude that the MS algorithms are the best decoding algorithms in terms of complexity and performance.

Table 1. Kintex 7 device overview for MS resources.

Summary of Device Utilization			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
No. of Slice Registers	2,599	407,600	1%
No. used as Flip Flops	2,599		
No. of Slice LUTs	5,813	203.800	2%
No. used as logic	5,284	203.800	2%
No. used as Memory	362	64,000	1%
No. of LUT Flip Flop pairs used	7,047	203.800	
No. of slice register sites lost to control set restrictions	254	407,600	1%
No. of bonded IOBs	4	500	1%
No. of BUFG/BUFGMUXs	9	890	1%
No. of BUFG/BUFGCTRLs	1	32	3%
No. of DSP48A1s	154	840	18%

Table 2. Comparison of resource devices.

Devices utilization	**Algorithms**	
	Log Domain	**Min-Sum**
Slice regester	3,950	2,599
LUT slice	21,616	5,813
Memory	362	362
DSP	193	154
Max-Delay (ns)	26.278	25.088

The result of a Xilinx SG simulation test between a binary broadcast Bernoulli signal and the received signal, where the MS decoder records the original signal with a 30ns delay, is shown in Fig. 19.

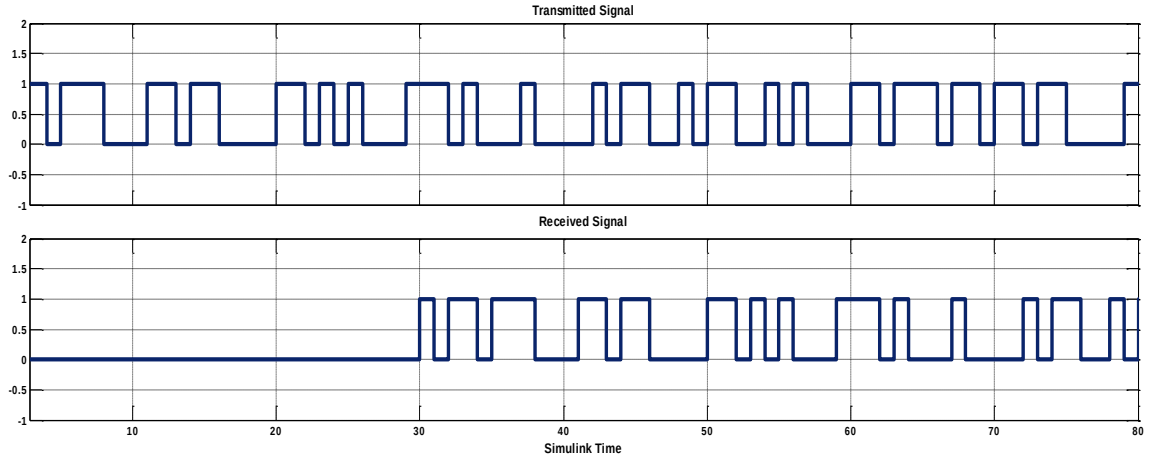

Fig. 19. The simulated output of the MS decoder algorithm's data recovery signal processing.

The user data information and recoverable information hardware Co-Simulation outcomes for the suggested technique are displayed in Fig. 10, with a delay due to the extraction process operation and some Xilinx blocks that cause delay.

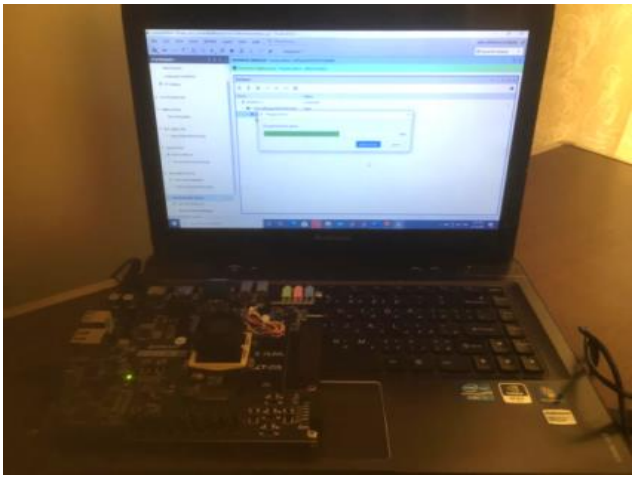

Fig. 20. The result of the installation of MS hardware Co-Simulation.

In this paper, it has been investigated a reduced reduction in resource utilization by 2.920476% which is the main contribution of this paper. On the other hand, the power consumption is out of the scope of the current paper it can be calculated in future work. In addition, the experiment is applied for the data rate (20) also it can be applied to the author's number of data in order to evaluate the performance of the current paper in future work.

4. CONCLUSION

This paper has suggested a low-complexity decoding algorithm for LDPC codes. The MS decoder algorithm is an improved version of the Log Domain algorithm. Simulations have been performed to analyze the BER efficiency of the communication system, Xilinx SG tools

were used to implement the design on the Kintex 7 FPGA development kit. According to the simulation results, MS was able to boost the Log Domain decoder's gain by 0.5dB. The MS algorithm used 1% of the slice registers, 18% of the DSP, and 2% of the LUT on the FPGA, whereas the other soft-decision decoders algorithms (Prob. Domain, Log Domain, and MS) have high resource utilization complexity, implying that the MS algorithm is the best decoding algorithm due to its BER performance and resource utilization. The suggested system's hardware implementation demonstrates that it is well-suited for future communication, particularly in real-time applications.

5. REFERENCES

Ahmed A.Emran, and Elsabrouty, Maha. "Simplified variable-scaled min sum LDPC decoder for irregular LDPC codes." *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*. IEEE, 2014.

Andrews, Kenneth S., et al. "The development of turbo and LDPC codes for deep-space applications." *Proceedings of the IEEE* 95.11 (2007): 2142-2156.

Angarita, Fabian, et al. "Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor." *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.7 (2014): 2150-2158.

Ceroici, Chris, and Vincent C. Gaudet. "FPGA implementation of a clockless stochastic LDPC decoder." *2014 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2014.

Chen, Jun-Chieh, et al. "A novel broadcast scheduling strategy using factor graphs and sum-product algorithm." *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*. Vol. 6. IEEE, 2004.

Hatami, Homayoon, et al. "A threshold-based min-sum algorithm to lower the error floors of quantized LDPC decoders." *IEEE Transactions on Communications* 68.4 (2020): 2005-2015.

Jianguang Zhao, Zarkeshvari Farhad, and Banihashemi Amir H., "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes." *IEEE transactions on communications* 53.4 (2005): 549-554.

Jun Heo, and Chugg Keith M. "Optimization of scaling soft information in iterative decoding via density evolution methods." *IEEE transactions on communications* 53.6 (2005): 957-961.

Johnson, Sarah J. "Introducing low-density parity-check codes." *University of Newcastle, Australia* 1 (2006): 2006.

Kalsi, Arvinder Pal Singh, Rajesh Guide Khanna, and Sanjay Guide Kumar. *SMT-8036 based implementation of secured adaptive Software Defined Radio system for LDPC coded modulation techniques*. Diss. 2012.

MacKay, David JC, and Radford M. Neal. "Near Shannon limit performance of low-density parity check codes." *Electronics letters* 32.18 (1996): 1645-1646.

MacKay, David JC. "Good error-correcting codes based on very sparse matrices." *IEEE transactions on Information Theory* 45.2 (1999): 399-431.

Mooij, Joris M., and Hilbert J. Kappen. "Sufficient conditions for convergence of the sum–product algorithm." *IEEE Transactions on Information Theory* 53.12 (2007): 4422-4437.

Mosleh, Mahmood Farhan. "Evaluation of low density parity check codes over various channel types." *Engineering and Technology Journal* 29.5 (2011): 961-971.

Mosleh, Mahmood F., Fadhil S. Hasan, and Aya H. Abdulhameed. "Enhancement of Differential Chaos Shift Keying Communication System Based on LDPC Codes." (2020).

Richardson, Tom, and Rudiger Urbanke. "The renaissance of Gallager's low-density parity-check codes." *IEEE Communications Magazine* 41.8 (2003): 126-131.

Roh, Si-Dong, Keol Cho, and Ki-Seok Chung. "Implementation of an LDPC decoder on a heterogeneous FPGA-CPU platform using SDSoC." *2016 IEEE Region 10 Conference (TENCON)*. IEEE, 2016.

Shannon, Claude Elwood. "A mathematical theory of communication." *ACM SIGMOBILE mobile computing and communications review* 5.1 (2001): 3-55.

Tomlinson M., Tjhai C.J., Ambroze M.A., Ahmed M., Jibril M. (2017) LDPC Codes. In: Error-Correction Coding and Decoding. Signals and Communication Technology. Springer, Cham. https://doi.org/10.1007/978-3-319-51103-0_12

Wiberg, N. Codes and Decoding on General Graphs. Linkoping, 1996. Dissertation. Linkoping University. Supervisor prof. Ingemar Ingemarsson.