Design and implementation of log domain decoder

Mahmood Farhan Mosleh[1], Fadhil Sahib Hasan[2], Ruaa Majeed Azeez[3]
[1,3]Department of Computer Engineering Techniques, Middle Technical University, Baghdad, Iraq
[2]Department of Electrical Engineering, College of Engineering, Al Mustansiriyah University, Baghdad, Iraq

Article Info

Article history:

Received Jul 18, 2019
Revised Oct 16, 2019
Accepted Oct 26, 2019

Keywords:

FPGA
LDPC
Log Decoder
Soft Decision
Xilinx system generator

ABSTRACT

Low-Density-Parity-Check (LDPC) code has become famous in communications systems for error correction, as an advantage of the robust performance in correcting errors and the ability to meet all the requirements of the 5G system. However, the mot challenge faced researchers is the hardware implementation, because of higher complexity and long run-time. In this paper, an efficient and optimum design for log domain decoder has been implemented using Xilinx system generator with FPGA device Kintex7 (XC7K325T-2FFG900C). Results confirm that the proposed decoder gives a Bit Error Rate (BER) very closed to theory calculations which illustrate that this decoder is suitable for next generation demand which needs a high data rate with very low BER.

Corresponding Author:

Mahmood Farhan Mosleh,
Department of Computer Engineering Techniques,
Electrical Engineering Technical College,
Middle Technical University, Baghdad, Iraq.
Email: drmahmoodfarhan@gmail.com

1. INTRODUCTION

LDPC codes were known for the first time by Gallager in 1960s [1], they belong to linear block codes that use generator matrix *G* in the encoding process and Parity Check Matrix (PCM) *H* for the decoding. The codes of LDPC can give sufficient reliability and at the same time, they approach performance very near to Shannon limit with reasonable complexity [2]. They become preferred in wireless communication standards such as Digital Video Broadcasting–Satellite–Second Generation (DVB-S2), WLAN (IEEE 802.11n) and WiMAX (IEEE 802.16e) because of their easy decoding process and improved error-correcting capability. They may be described by sparse PCM. In this PCM the ones entries are very low as compared to the zero entries [3, 4] which enable encoding and decoding with low complexity. Also, the Tanner graph can be used to represent LDPC codes. Check nodes (CNs) can be used to represent every row, and Variable Nodes (VNs) can be used to represent every column. The 1s entries in the matrix denote the links between the CNs and VNs [5].

The encoding of these codes is done by using either the generator matrix method in which the message is encoded by multiplying the massage by the *G* matrix [6] or by using the approximate lower triangulation. The idea of this method is to encode a message using the PCM instead of the *G* matrix. Where the matrix must be converted into an approximate lower triangular form in order to make the encoding process [7].

Many researches were interesting in implementing LDPC codes via FPGA. Like [8] where a novel design in the decoding of LDPC codes was presented. This design used a Progressive Edge Growth (PEG) algorithm using a regular code and an improved Min-Sum (MS) decoding algorithm that led to improving the performance with no further hardware overhead. The author in [9] described the execution of a common and embedded decoder to evaluate the unstructured LDPC system under Additive-White Gaussian Noise (AWGN) channel. This technique of Hardware/Software implementation offered the maximum flexibility for the improvement and fast prototyping of the hardware-based simulator system. Also, the authors in [10, 11] proposed a modified design of decoders which reduced the complexity of decoding LDPC codes based on FPGA. In addition, the author in [12] implemented the LDPC decoder based Xilinx System Generator (XSG). Furthermore, the authors in [13] designed and implemented LDPC decoder based FPGA which reduced the complexity that can be used for applications of high data rate. Moreover, the authors in [14-18] used FPGA technique to implement LDPC decoders with different algorithms. Finally, the author in [19] proposed the FPGA construction of spatially coupled (SC) LDPC codes resulting from quasi-cyclic (QC) LDPC codes.

In this paper, a design of LDPC system using the log domain algorithm will be presented using Xilinx System Generator (XSG) which is a new and efficient technique to design many systems such as in [20-22]. The LDPC system will be implemented using XSG with the software tools Xilinx Vivado 2017.4 and Xilinx Kintex7 (XC7K325T-2FFG900C). The purpose of this paper is to design a communication system using LDPC code based soft decision decoder represented by the log domain algorithm with reduced hardware design to assess its performance and to determine the extent to which the theoretical results match the current application. The rest of this paper, the second section includes the theory of log domain algorithm used in the decoder, the third section includes details of the design using XSG. In the fourth and fifth sections, the XSG waveforms and synthesis reports are presented in these sections respectively. In the sixth section, a conclusion is presented.

2. LDPC BASED ON LOG DOMAIN DECODER SYSTEM

Figure 1 shows the block diagram of LDPC system using the log-domain decoder.

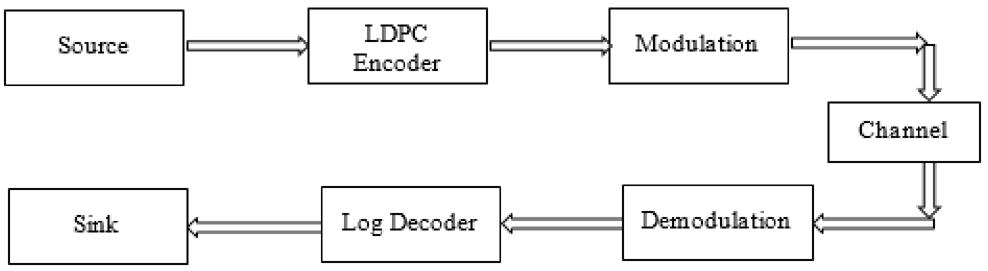

Figure 1. LDPC System model based on log decoder

The LDPC encoder is implemented using the matrix method with n is 20 and m is 10 by computing the check equations of the matrix in the systematic form $H_{sys} = [I_m, A_{m\times K}]$ where k= n-m as shown below:

$$H_{sys} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&1&1&1&1&0 \\ 0&0&0&1&0&0&0&0&0&0&1&1&0&1&0&0&0&0&1&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&1&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&0 \end{bmatrix} \quad (1)$$

Where the PCM consists of identity matrix I which represents the check equations and binary matrix A which represents the information part by depending on the binary matrix the check equations can be computed as follows:

$$C_1 = d_3 + d_6 + d_9 \quad (2)$$

and

$$C_2 = d_5 + d_6 + d_7 + d_8 \quad (3)$$

Where C_1 and C_2 represent the equations for first and second row respectively and so on for the other equations and d_i represent the data bits where i is from 1 to 10. The details description of log decoder is explained below.

2.1. Log decoder

The log domain algorithm is a type of Sum-Product Algorithm (SPA) which depends on passing messages between CNs and Bit Nodes (BNs). It is like the Bit Flipping Algorithm (BFA) but the BFA takes a prior hard decision on the received bits as inputs whereas the log domain algorithm depends on a soft decision by taking the probability of every receiving bit as input [23]. For easier computation for SPA, the log-likelihood ratio (LLR) of prior (which represent the receiving messages from the channel) and posterior (which represent the medial messages moved between CNs and VNs) probability is used. The process of the decoding contains three steps which are the initializing step, CNs processes and VNs processes [24]. These steps are listed below [25]:

– Step1 (Initialization): The prior messages sent from BN n to the CN m represent the LLR.

$$Lci_n = \frac{-4r_i}{N_0} \tag{4}$$

$$alphaij_{m,n}=\text{sign}\ (Lci_n) \tag{5}$$

$$betaij_{m,n} = |Lci_n| \tag{6}$$

where Lci_n denotes the LLR of the nth bit in the mth parity-check with for an AWGN channel with Signal to Noise Ratio (SNR), r_i is the received signal from the channel *and* N_0 is the noise variance, $alphaij_{m,n}$ and $betaij_{m,n}$ will calculate the sign and the absolute value to each Lci_n.

– Step2 (CN-to-BN messages): Computing the extrinsic messages for each set of bits connected to CN m by excluding the bit n'.

$$Pibetaij_{m,n} = \log\left(\frac{1+\prod_{n\in B_m, n'\neq n} \tanh\left(\frac{betaij_{m,n'}}{2}\right)}{1-\prod_{n\in B_m, n'\neq n} \tanh\left(\frac{betaij_{m,n'}}{2}\right)}\right) \tag{7}$$

where $Pibetaij_{n,m}$ represents the probability that parity-check m is satisfied if bit n is supposed to be a 1 for the LLR.

– Step3(Codeword test):

$$L_{Qi} = Lci_n + \sum_{m\in A_n} Pibetaij_{m,n} \tag{8}$$

where L_{Qi} represents the collective log-likelihood ratio for nth digit, is the addition of the extrinsic messages and the original LLR that was calculated in the first Step.

For every bit a hard decision will be done:

$$vhat = \begin{cases} 1, & L_{Qi} < 0 \\ 0, & L_{Qi} > 0. \end{cases} \tag{9}$$

If *vhat* a valid codeword ($Hvhat^T = 0$), or if the maximum iteration's number are ended, the algorithm will be terminated.

– Step4(BN-to-CN messages)*:* The message sent from every BN n to CN m that it is connected to, it is similar to (8), except that bit n sends to CN m a LLR computed without using the information from CN m' [25]:

$$Lqij_{m,n}= Lci_n + \sum_{m'\in A_n, m'\neq m} Pibetaij_{m',n} \tag{10}$$

3. XSG IMPLEMENTATION OF LDPC CODE BASED ON LOG DECODER

The system in Figure 2 is implemented using XSG. All blocks are corresponding to the blocks in Figure 1**.** The detail description of each XSG component is presented:

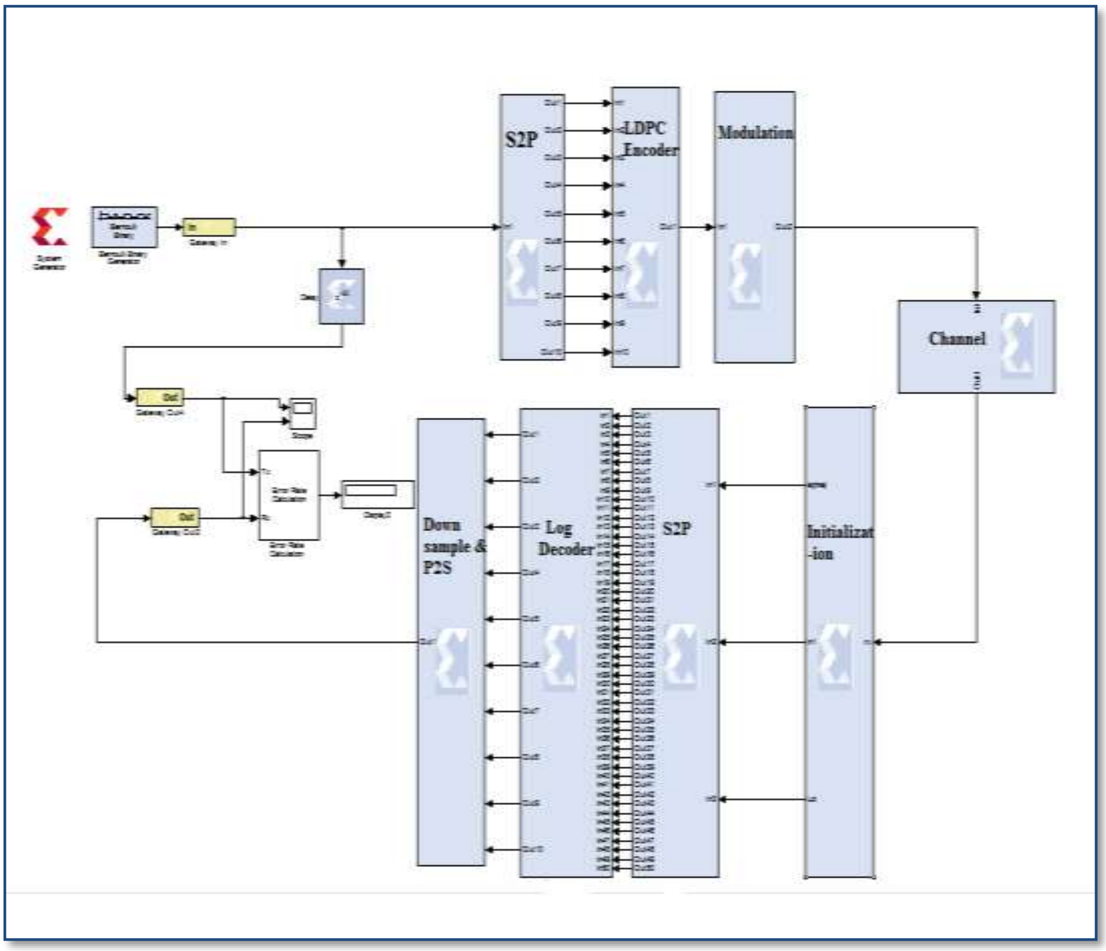

Figure 2. LDPC system using XSG

3.1. Transmitter section

The transmitter section consists of source input, serial to parallel, LDPC encoder and modulation. Each one of these blocks is designed using XSG in Simulink/Matlab program.

3.1.1. Source input

The block of Bernoulli Binary Generator has been used to generate the Bernoulli Binary Signal. The performance of the suggested system has been tested through using the binary data. The setting of this block is as the following: the sample time is 1 and the format of the resulting data is Boolean. The gateway-in has been used in order to convert the data to unsigned format with Word Length (WL) is 1 and Fraction Length (FL) is 0. The output from this block will be the source input to the LDPC encoder.

3.1.2. Serial to parallel block(S/P)

This block is existing in XSG tools. The bit's number is 10 in this block and 1 is used for latency. The resulting from this block is a symbol of 10 bits which can be converted to parallel bits. The slice block has been used to choose a particular sort of bits from every sample in the input. There are ten slice blocks to extract the ten bits. The number of bits in each slice is 1 and the format of the resulting data is unsigned with WL is 2 and FL is 0. The block diagram of S/P in XSG is shown in Figure 3**.**

3.1.3. LDPC encoder

The LDPC encoder consists of xor gates and concat block which are available in XSG library. The generator matrix method has been used for encoding each message. Each xor gate denotes the parity check equation for every row of the *H* matrix in (1). The output of these gates will then enter to the concat block to get a symbol of 20 bits which represent the encoded message. The block diagram of LDPC encoder in XSG is shown in Figure 4.

3.1.4. Modulation

The modulation block consists of parallel to serial block and mapping block. Every bit from the serial data has represented as an address to the ROM to indicate either the phase of $\{0^0\}$ or the phase of $\{180^0\}$. The initial value vector for the ROM is [-1 1]. The format of the resulting data will be signed with WL is two bits and FL is zero bit. Then delay has been linked for enabling pin of the ROM block to mask all the serial bits till they are ready for map process. The delay is used here for synchronizing between the current bit and the previous bit. The XSG block of modulation is shown in Figure 5.

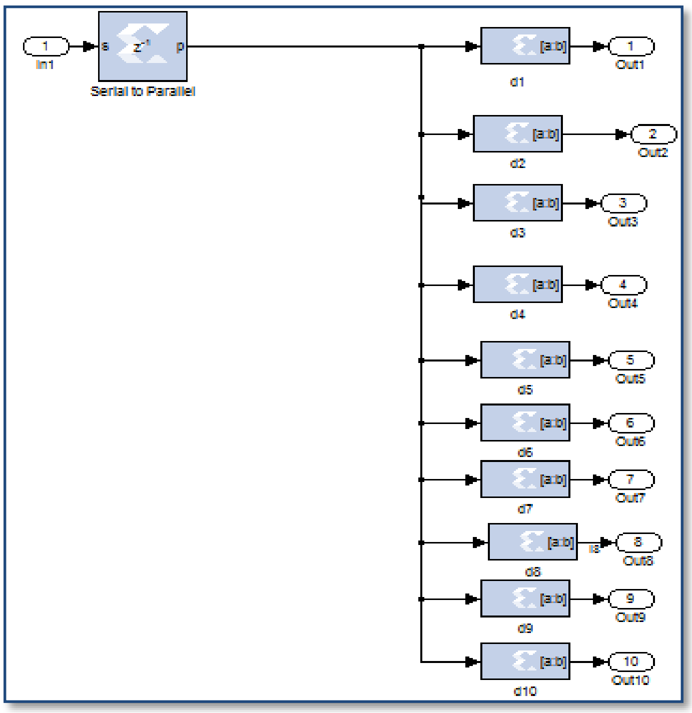

Figure 3. S/P converter in XSG

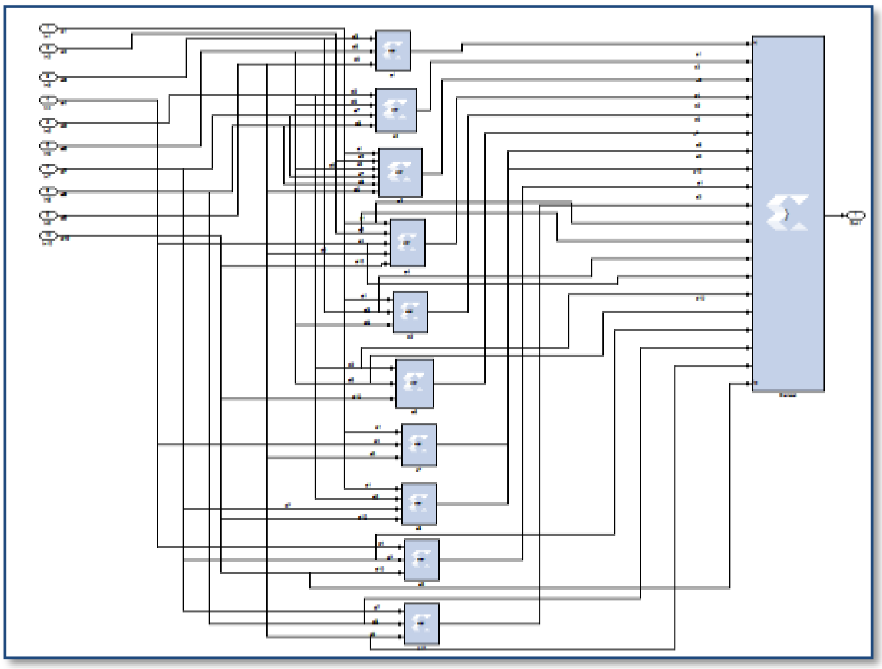

Figure 4. Block diagram of LDPC encoder in XSG

3.2. AWGN channel

The AWGN channel has been used for testing the performance of the system, where random signals are added with the received signals. The AWGN generator is a system generator block set used to generate random signals with Gaussian distribution of zero mean and unity variance with WL is 18 bits and FL is 16 bits. The output of AWGN is multiplied by a constant that represents the square root of N_0 then the result from this multiplication will be added with each bit of the transmitted signal to represent the noise for different SNR. The XSG block diagram of AWGN channel is illustrated in Figure 6.

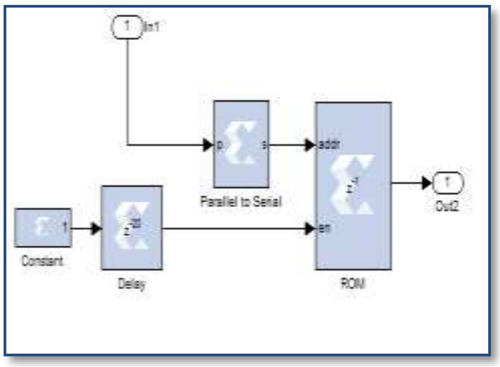

Figure 5. Block diagram of modulation in XSG

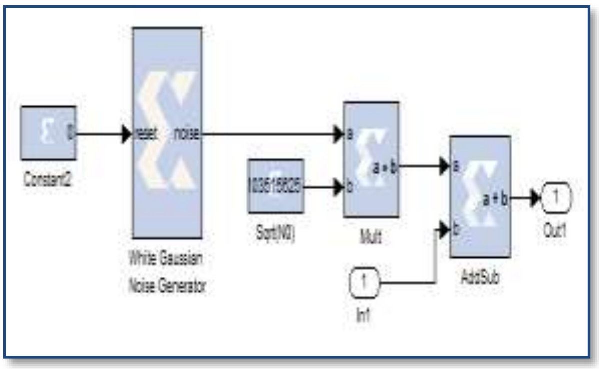

Figure 6. AWGN channel in XSG

3.3. Receiver part

The receiver section consists of the following blocks:

3.3.1. Initialization

Initialization is implemented by multiplying each bit by the value of -4/N_0, where N_0 is the noise variance for each Signal to Noise Ratio (SNR) according to (4). For SNR from 0 to 7 the values of N_0 are 1, 0.7943, 0.6310, 0.5012, 0.3981, 0.3162, 0.2512 and 0.1995 respectively. This process will be applied to 20 bits serially as shown in Figure 7. After this, the sign value will be computed for each bit beside the absolute value in order to apply (7) to each bit as shown in Figure 8.

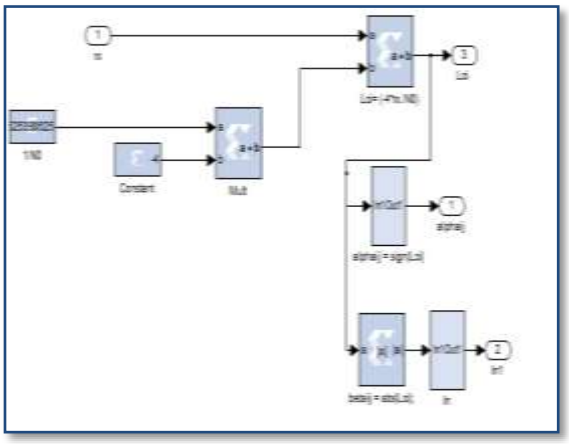

Figure 7. XSG block diagram of initialization

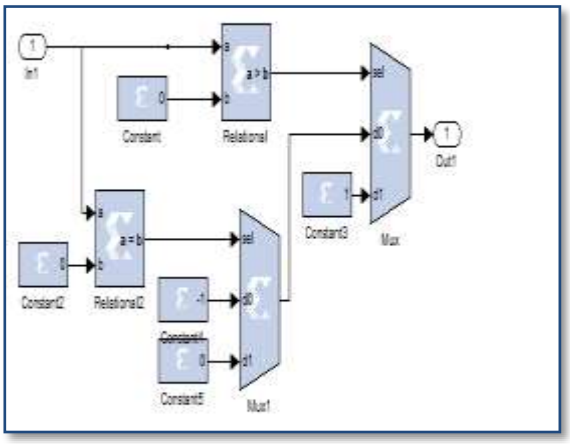

Figure 8. XSG block diagram of alphaij block

3.3.2. Serial to parallel

Serial to parallel block has been used for converting the serial samples to parallel samples, where each sample is with format WL is 18 bits and FL is 16 bits and 20 latches registers have been used to mask the parallel 20 samples. This block has been implemented using shift registers. The block diagram of serial to parallel in XSG is shown in Figure 9

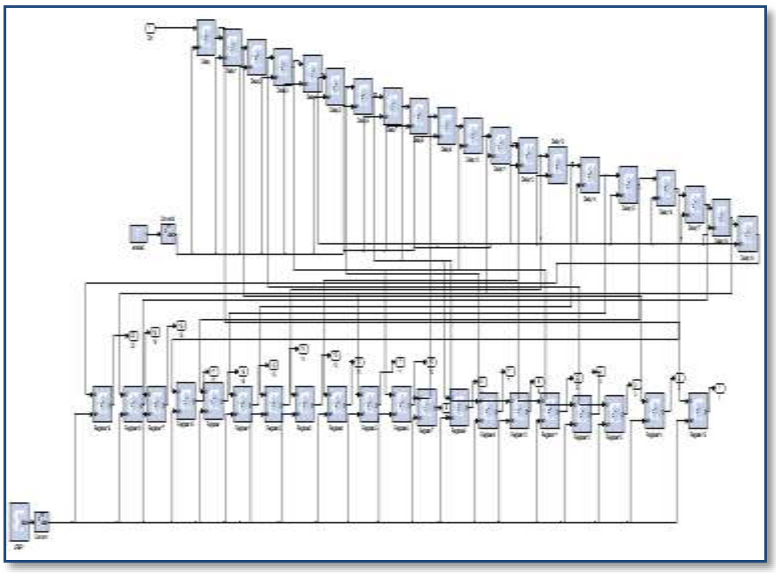
Figure 9. Block diagram of serial to parallel in XSG

3.3.3. Log Decoder

The inputs to this block are symbols of 20 bits and the outputs will be symbols of 10 bits which represent the information message after the decoding process as shown in Figure 10. The blocks in part A represent the horizontal step and the blocks in part B represent the vertical step.

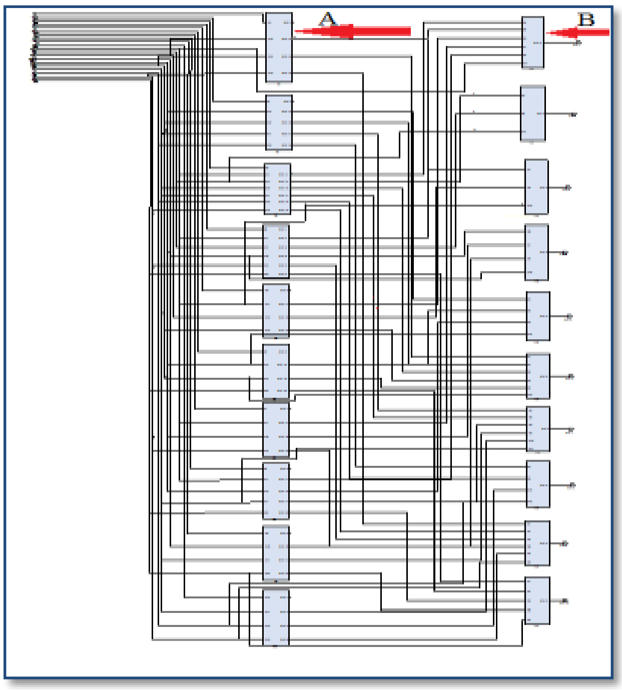

Figure 10. Block diagram of log decoder in XSG

3.3.3.1 Horizontal Step

In this step, the computations are made based on the number of 1s in every column of the ten rows in the PCM. The extrinsic probabilities for each bit are computed by multiplying the sign value for each bit by the summation of the probabilities of non-zero bits in each row. The computations will depend on the number of non-zero columns in every row as shown in Figure 11.

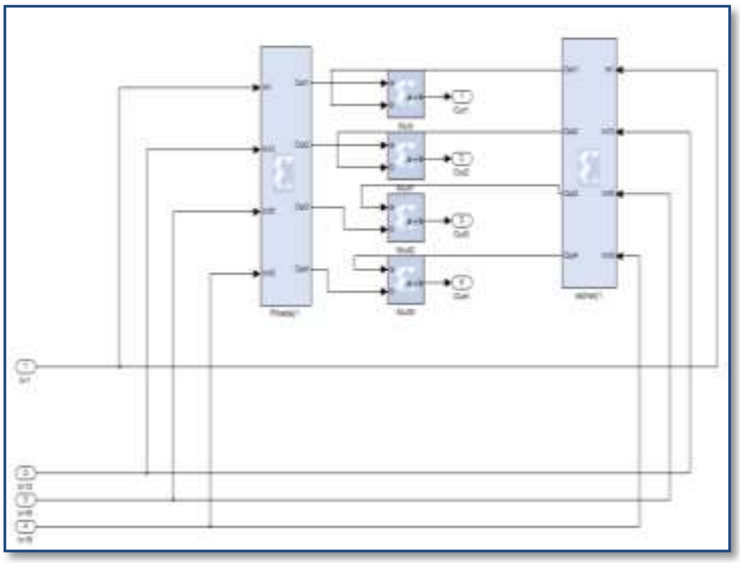

Figure 11. Extrinsic probabilities for row1

3.3.3.2. Vertical Step

In this step, bit messages have been updated depending on the summation of the extrinsic probabilities for each bit with the LLR without using the information from CN m', also the decision is made for each bit by combining the extrinsic probabilities for every bit and the LLR from the channel. The decision for decoding each bit based on its value if it is positive or negative if the value is negative the bit is one otherwise is zero as shown in Figure 12.

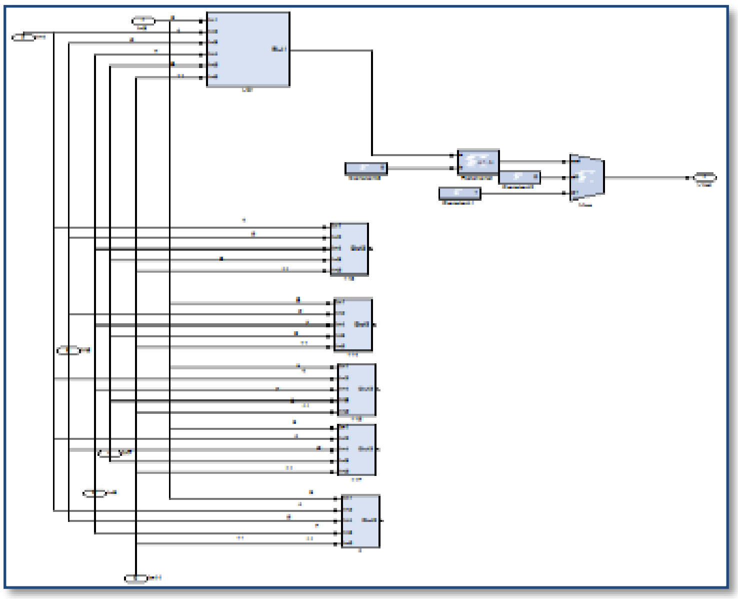

Figure 12. Decoding decision for column 11

3.3.4 Down Sample and Parallel to serial

The resulting symbols from the decoding process which represent the information part will then enter to the down sample block which is implemented for decreasing the rate of the signal at the receiver. The parameters for this block are sample rate is 20 and latency is one. After this stage, these bits will be transformed from parallel data to serial data by using concat block and parallel to serial block as shown in Figure 13.

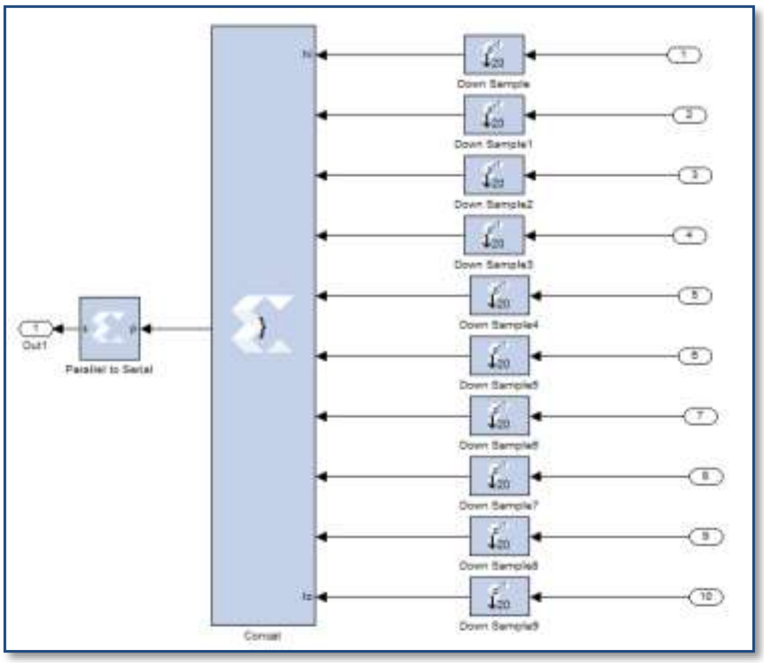

Figure 13. XSG block diagram of down sample block with concat and parallel to serial

4. RESULTS AND ANALYSIS

The XSG simulation waveforms for each block of the system model is plotted using Matlab program. Figure 14 shows XSG time waveforms of S/P block. The time representation of LDPC encoder and the modulation block are depicted in Figures 15 and 16 respectively. Figures 17 shows the time waveform of the output of the AWGN channel. Figure 18 shows XSG time waveforms of the initialization process. Figure 19 shows XSG time waveforms of the decoding and down sample for the first bit. Figure 20 shows XSG time waveform of the concat block output. The comparison between XSG time waveform of the transmitter and receiver is shown in Figure 21. Table 1 illustrates the BER results comparison between Matlab/simulation and XSG.

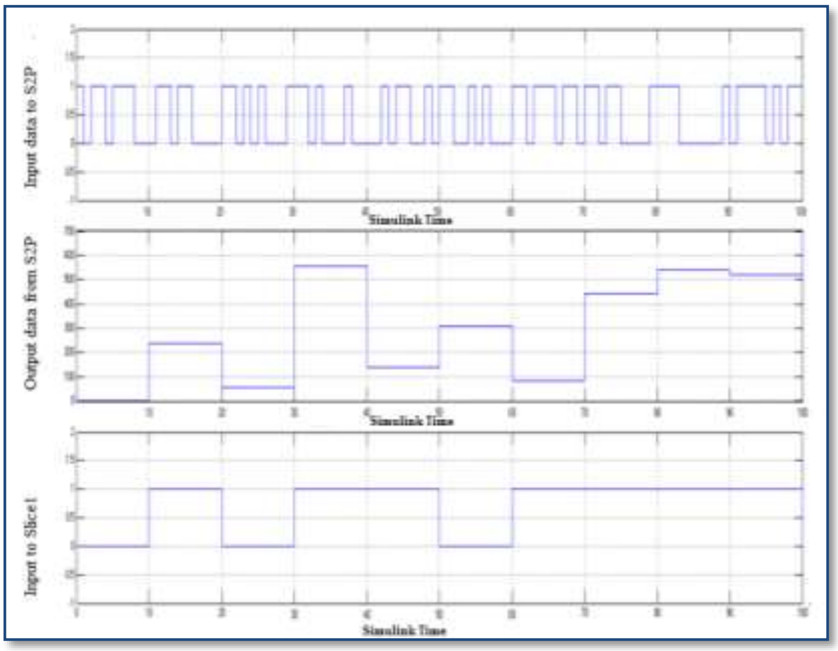

Figure 14. XSG time waveforms of the S/P

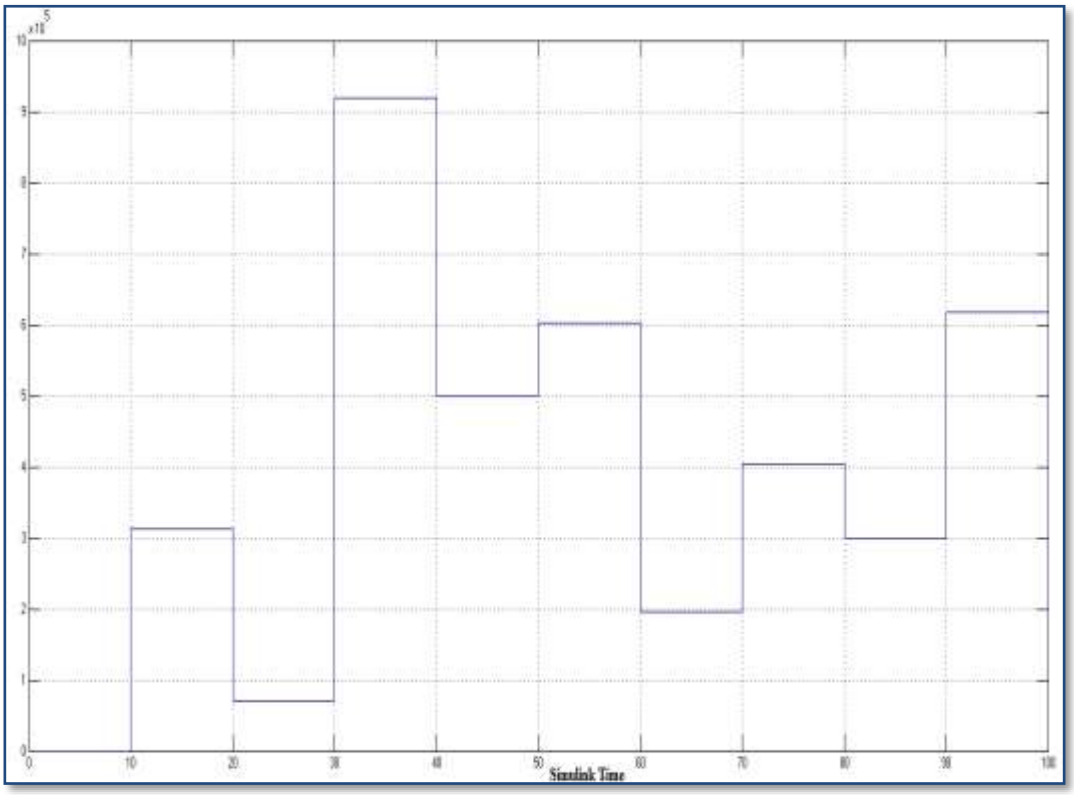

Figure 15. XSG time waveform of the LDPC encoder output

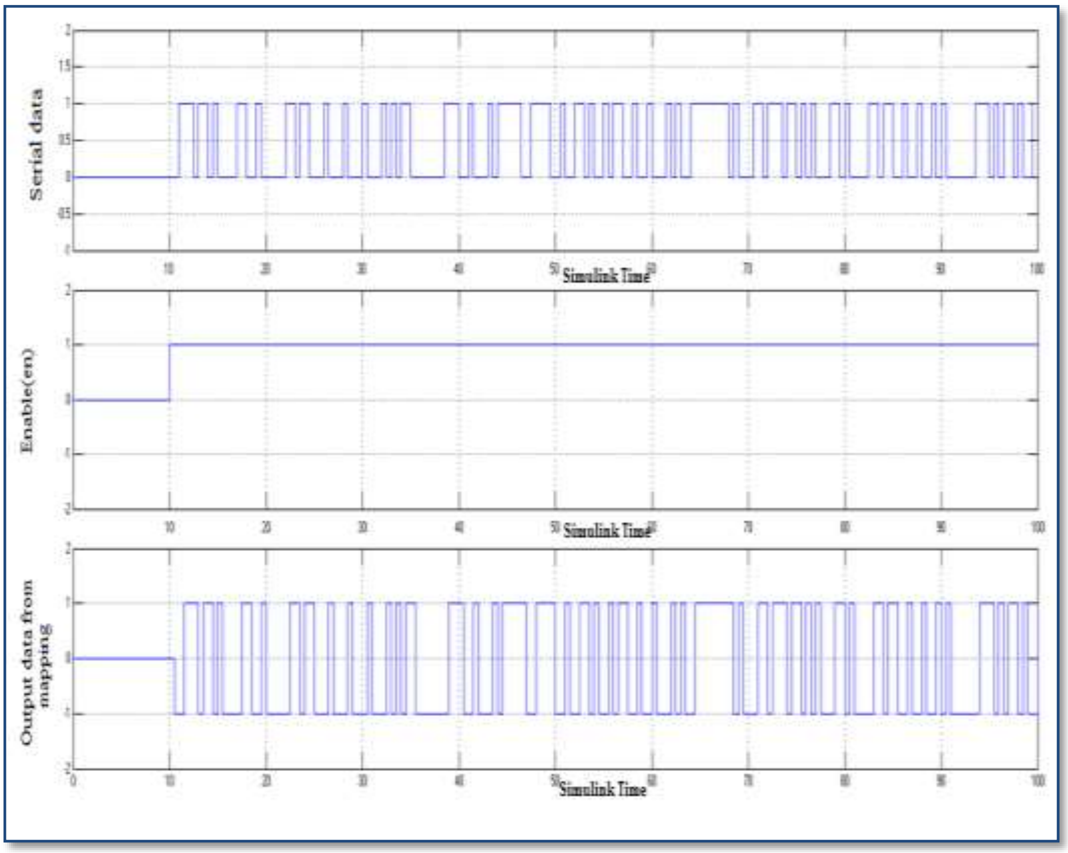

Figure 16. XSG time waveforms of the modulation block

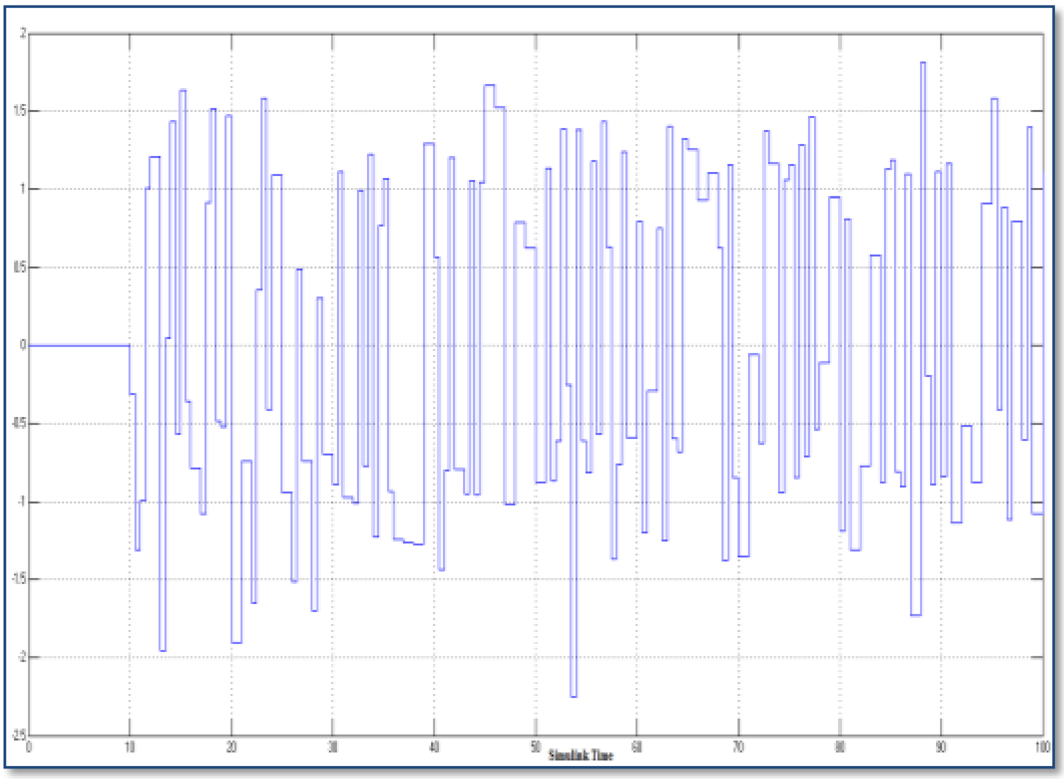

Figure 17. XSG time waveform of the AWGN channel

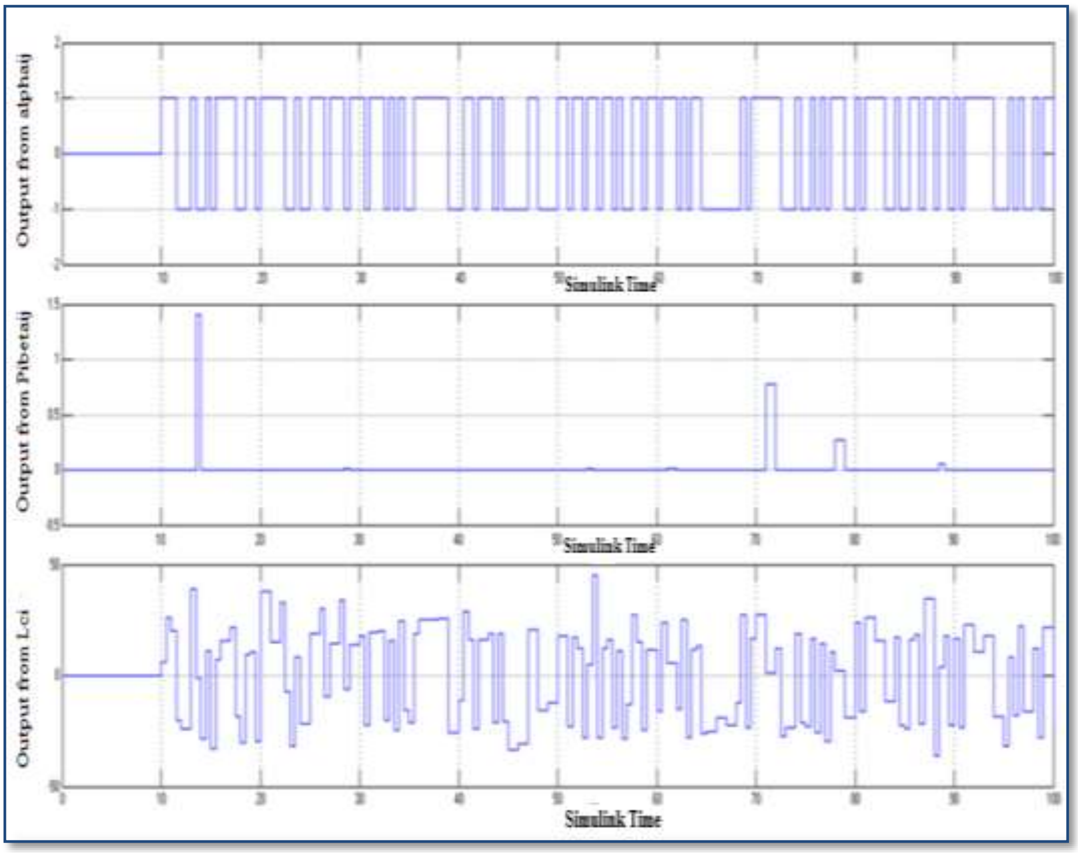

Figure 18. XSG time waveforms of the initialization process

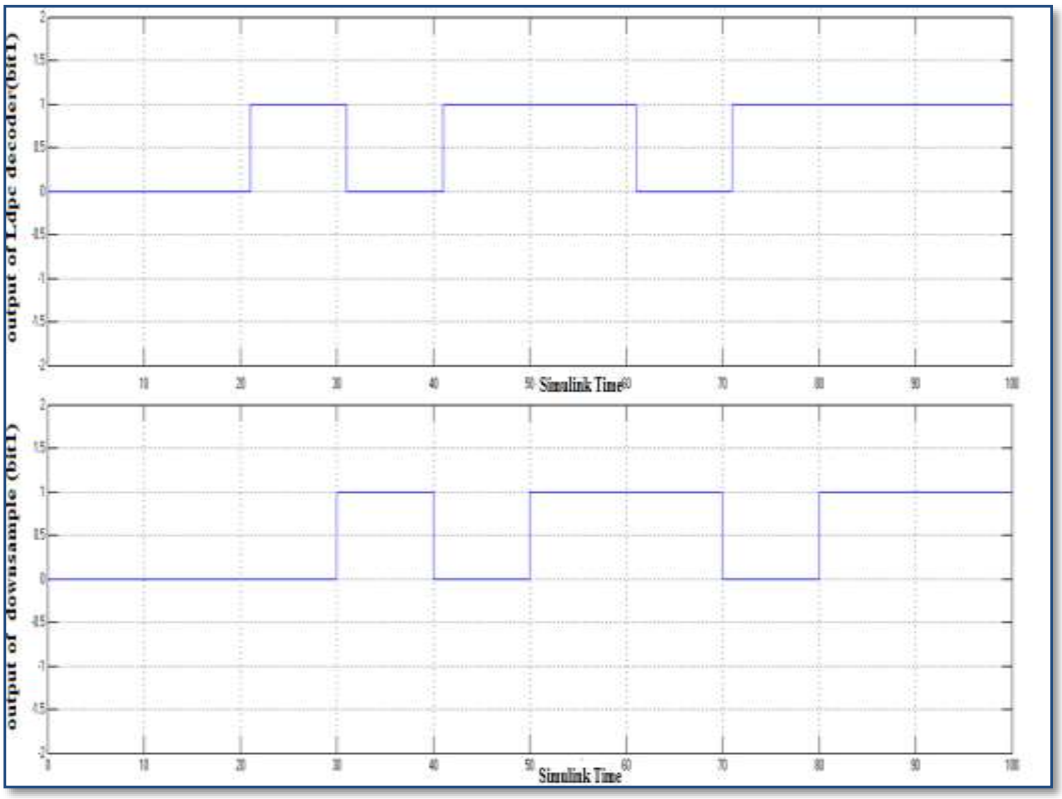

Figure 19. XSG time waveforms of the decoding and down sample for the first bit

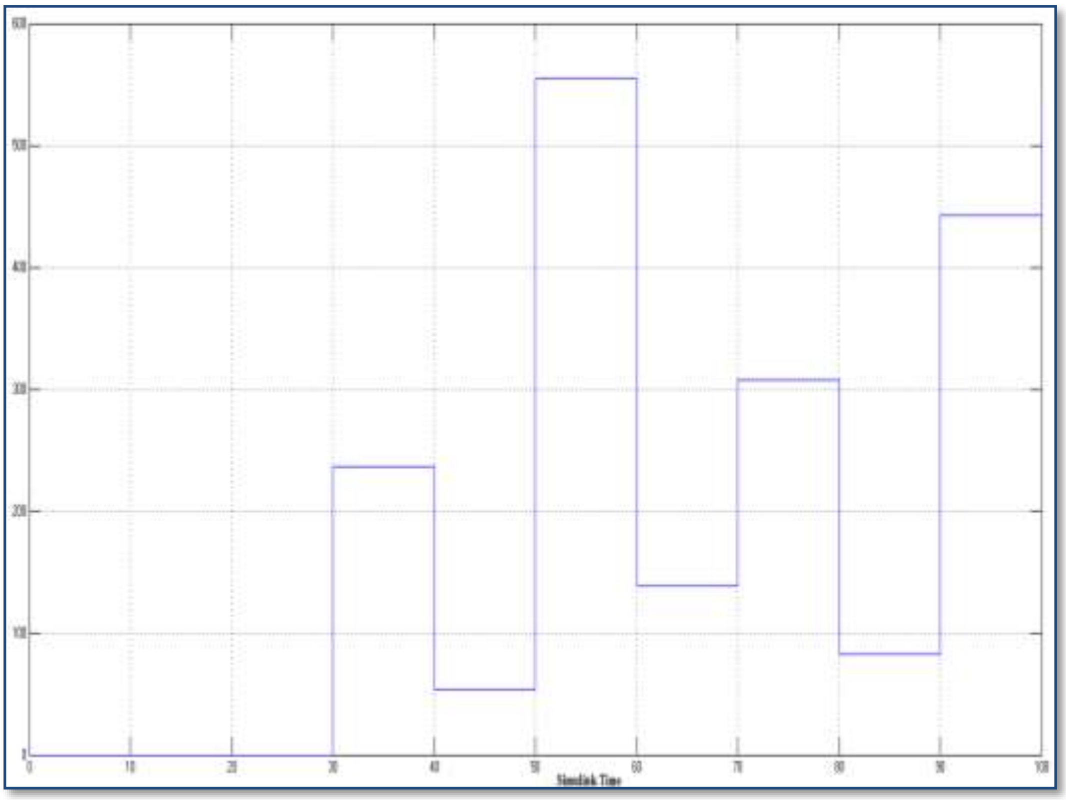

Figure 20. XSG time waveform of the concat block output

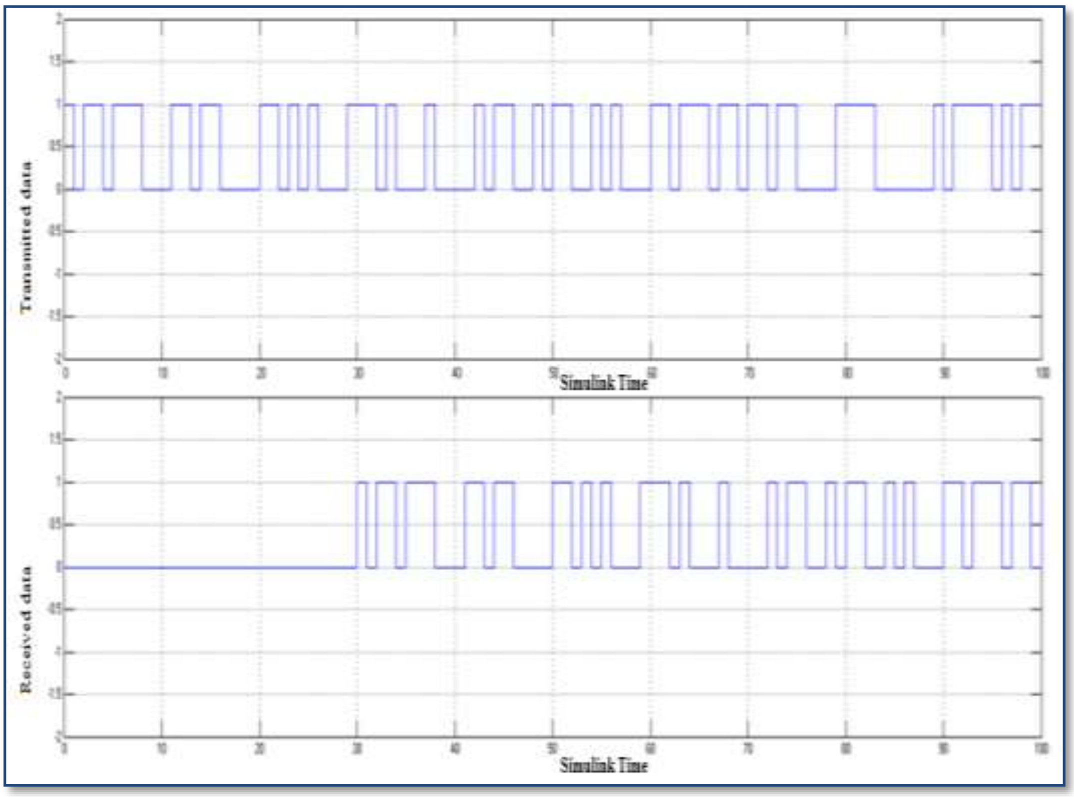

Figure 21. Comparing the signal at the transmitter and at the receiver

Table 1. Comparing the BER in Matlab/simulation and BER in FPGA

SNR(dB)	BER simulation	BER FPGA
0	0.1429	0.1089
1	0.1000	0.09901
2	0.0786	0.06931
3	0.0286	0.0396
4	0.0429	0.0495
5	0	0.009901
6	0	0.009901
7	0	0

5. SYNTHESIS REPORTS

The Vhdl code can be generated by choosing the FPGA device of kintex7 (XC7K325T-2FFG900C) and Vivado 2017.4 program. The device utilization summary of LDPC system is presented in Table 2. The minimum period is 26.278ns and the maximum frequency is 38.055MHz. In addition, the proposed LDPC system based on log domain decoder has been tested and downloaded successfully on the FPGA device kintex7 (XC7K325T-2FFG900C) as shown in Figure 22.

Table 2. Device utilization summary of log decoder system

Resource	Utilization	Available	Utilization Percentage
LUT	44120	203800	21.65
LUTRAM	253	64000	0.40
FF	3571	407600	0.92
BRAM	4.50	445	1.01
DSP	193	840	22.98
Number of bonded IOBs	4	500	0.80
BUFG	1	32	3.13

Figure 22. Test the system using kintex7 (XC7K325T-2FFG900C)

6. CONCLUSION

A complete LDPC decoder system based log domain algorithm has been designed and implemented successfully by using XSG. The software tools Xilinx Vivado 2017.4 and kintex7 (XC7K325T-2FFG900C) have been used in the implementation. The obtained results from the system proved that the system works correctly with results very close to theoretical results. The results confirm that such decoder can be applied with a new communication system according to its low complexity and low BER. FPGA results for BER were 0.1089, 0.09901, 0.06931, 0.0396, 0.0495, 0.009901, 0.009901 and zero for SNR values from 0, 1, 2,….,7 respectively. The worst BER is achieved when the SNR value is zero while the best BER is achieved when the SNR value is 7. This is mean increasing the SNR value will lead to significant improvement in BER due to the increase in signal power over the noise power. The results confirm that the theoretical results based on pure simulation are very close to those results of FPGA implementation as indicated in Table 1. That means the proposed system can be implemented successfully in modern communications.

REFERENCES

[1] R. G. Gallager, "Low Density Parity Check Codes", Number 21 in Research monograph series. MI Press, Cambridge, Mass, 1963.

[2] M. K. Roberts and R. Jayabalan, "A Modified Normalized Min – Sum Decoding Algorithm for Irregular LDPC Codes", *International Journal of Engineering and Technology*, Vol. 5, No. 6, Jan 2014.

[3] N.P. Bhavsar and B. Vala, "Design of Hard and Soft Decision Decoding Algorithms of LDPC", *International Journal of Computer Applications*, Vol. 90, No.16, Mar 2014.

[4] M. F. Mosleh, "Evaluation of Low Density Parity Check Codes Over Various Channel Types", *Engineering and Technology Journal*, Vol. 29, Issue 5, pp. 961-971, 2011.

[5] Myung Hun Lee, Jae Hee Han and Myung Hoon Sunwoo, "New Simplified Sum-Product Algorithm For Low Complexity LDPC Decoding", *2008 IEEE Workshop on Signal Processing Systems*, Washington, DC, pp. 61-66, 2008.

[6] O. O. Khalifa, et al., "Performance Evaluation of Low Density Parity Check Codes", *Journal of Electrical and Electronics Engineering*, Vol.2, No.6, Jun 2008.

[7] T. J. Richardson and R. L. Urbanke, "Efficient Encoding of low-density parity-check codes", in *IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 638-656, Feb 2001.

[8] R. Zarubica, S. G. Wilson and E. Hall, "Multi-Gbps FPGA-Based Low Density Parity Check (LDPC) Decoder Design", *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, Washington, DC, pp. 548-552, 2007.

[9] S. M. E. Hosseini, Kheong Sann Chan and Wang Ling Goh, "A reconfigurable FPGA implementation of an LDPC decoder for unstructured Codes", *2008 2nd International Conference on Signals, Circuits and Systems*, Monastir, pp. 1-6, 2008.

[10] V. A. Chandrasetty and S. M. Aziz, " FPGA Implementation of High Performance LDPC Decoder using Modified 2-bit Min-Sum Algorithm", *2010 Second International Conference on Computer Research and Development*, 2010.

[11] S. S. Khati, P. Bisht and S.C. Pujari, "Improved Decoder Design for LDPC Codes based on selective node processing", *2012 World Congress on Information and Communication Technologies*, 2012.

[12] A. K. Panigrahi and A. K. Panda, " Design of LDPC Decoder Using FPGA: Review of Flexibility", *International Journal of Engineering and Science*, Vol. 1, pp. 36-41, Nov 2012.

[13] J.C. Porcello, "Designing and Implementing Low Density Parity Check (LDPC) Decoders using FPGAs," Aerospace Conference, *IEEE,* 2014.

[14] T. Nguyen-Ly et al., "FPGA Design of High Throughput LDPC Decoder based on Imprecise Offset Min-Sum Decoding", *IEEE 13th International New Circuits and Systems Conference*, 2015.
[15] D. Zou and I. B. Djordjevic, "FPGA-based rate-adaptive LDPC-coded modulation for the next generation of optical communication systems", *Optics Express*, Vol. 24, No. 18, pp. 21159–21166, 2016.
[16] A. Boudaoud, M. El Haroussi, E. Abdelmounim, "VHDL Design and FPGA Implementation of LDPC Decoder for High Data Rate", *International Journal of Advanced Computer Science and Applications*, Vol. 8, No. 4, 2017.
[17] P.V Sreemohan and N. Sebastian, "FPGA Implementation of Min-Sum Algorithm for LDPC Decoder", *International Conference on Trends in Electronics and Informatics*, 2017.
[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, No. 9, pp. 1683–1690, 2017.
[19] X. Sun and I. B. Djordjevic, "FPGA implementation of rate-adaptive spatially coupled LDPC codes suitable for optical communications", *Optics Express*, Vol. 27, No. 3, 4 Feb 2019.
[20] Mahmood F. Mosleh, Mais F. Abid and Mohammed Al-Sadoon, "Implementation of Turbo Code Based Xilinx System Generator", *9th International EAI Conference*, Broadnets Faro, Portugal, 2018.
[21] Dong-U Lee," Hardware Designs for Function Evaluation and LDPC Coding", Ph.D. *Thesis*, Imperial College London, 2004.
[22] Asisa K. Panigrahi and Ajit K. Panda, "Design of LDPC Decoder Using FPGA: Review of Flexibility", *International Journal of Engineering and Science*, Vol. 1, Issue 7, pp. 36-41, 2012.
[23] S. J. Johnson, "Introducing Low Density Parity Check Codes", MSc. *Thesis*, University of Newcastle Australia, 2004.
[24] M. Kiaee, H. Gharaee and N. Mohammadzadeh,"LDPC Decoder Implementation Using FPGA", *8th International Symposium on Telecommunications*, 2016.
[25] M. Kolayli, "Comparison of Decoding Algorithms for Low-Density Parity-Check Codes", MSc. *Thesis*, Middle East Technical University, 2006.

BIOGRAPHIES OF AUTHORS

Mahmood F. Mosleh. He received his B.Sc, M.Sc and Ph.D degrees in 1995, 2000 and 2008 respectively. He has been a Prof. of Communication Eng. at the MTU, Iraq. He has more than 35 publications in national and international conferences and journals. Prof. Mahmood is a member of IEC and he is the head of Iraqi committee of IEC.

Fadhil S. Hasan was born in Baghdad, Iraq in 1978. He received his B.Sc. degree in Electrical Engineering in 2000 and his M.Sc. degree in Electronics and Communication Engineering in 2003, both from the Mustansiriyah University, Iraq. He received Ph.D. degree in 2013 in Electronics and Communication Engineering from the Basrah University, Iraq. In 2005, he joined the faculty of Engineering at the Mustansiriyah University in Baghdad. His recent research activities cover wireless communication systems, multicarrier system, wavelet based OFDM, MIMO system, speech signal processing, chaotic modulation, FPGA and Xilinx system generator based communication system**.** Now he has been an assistant Prof. at AL Mustansiriyah University, Iraq.

Ruaa Majeed Azeez was born in Iraq on March 11, 1989. She received her B.Sc. in Computer Engineering Techniques, Middle Technical University, Baghdad Iraq, in 2011. She worked as a teaching assistant in AL Furat AL Awsat Technical University since 2013. She is currently pursuing her MSc. program in Computer Technical Engineering with an emphasis in communication systems.