ISSN : 2350-0743

International Journal of Recent Advances in Multidisciplinary Research

Vol. 13, Issue 05, pp.12435-12448, May, 2026
https://doi.org/10.64485/ijramr.6559.05.2026

RESEARCH ARTICLE

MODEL-BASED DESIGN AND IMPLEMENTATION OF SDR TRANSCEIVER USING QC-LDPC CODING-DECODING OVER AWGN CHANNEL ON FPGA USING MATLAB/SIMULINK

***Rehan Muzammil**

Senior Member, IEEE, Department of Electronics Engineering, Aligarh Muslim University, Aligarh, India

ARTICLE INFO

Article History:
Received 15th February, 2026
Received in revised form
24th March, 2026
Accepted 19th April, 2026
Published online 28th May, 2026

Keywords:

AWGN, QAM, Low Density Parity Check (LDPC), Coder, Decoder, FPGA, Model-Based Design.

****Corresponding author:***
Rehan Muzammil

ABSTRACT

For the sixth-generation (6G) mobile communication, the channel coding scheme is very important. To implement high-speed low-density parity-check (LDPC) coders and decoders, Field-programmable gate arrays (FPGAs) are widely used. The well-known practice is to develop a register-transfer level (RTL) model of a digital circuit, which requires extensive simulation and system verification, resulting in a long development cycle. Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes are used to correct transmission errors in digital communication systems. Initially, ASICs were targeted due to computational complexity; many-core and multicore systems abundantly use LDPC decoders. This paper generates the code word using an 8x16 Generator Matrix derived from an 8x16 parity-check matrix. LDPC decoding is performed using hard decisions and a bit-flipping algorithm. The transceiver is modelled in MATLAB/Simulink and tested in real time on an FPGA. The FPGA board used is the Xilinx Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit. This board features a System-on-a-Chip (SoC) comprising an ARM processor and an FPGA. A part of the transceiver runs on the FPGA, and the rest runs on the ARM processor. Model-based design saves significant time by combining design, coding, and testing. The MATLAB/Simulink HDL Workflow advisor automatically generates code in C++ and VHDL.

Copyright©2026, Rehan Muzammil. *This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.*

Citation: Rehan Muzammil, 2026. *"Model-Based Design and Implementation of SDR Transceiver using QC-LDPC Coding-Decoding over AWGN Channelon FPGA Using Matlab/Simulink" , International Journal of Recent Advances in Multidisciplinary Research, 13,(05), 12435-12448.*

INTRODUCTION

Over the past few decades, the transition from analogue transmission methods to digital radio has been under discussion, and tests are underway in several countries. LDPC codes are frequently used for error correction. Both Gallager in 1960 and Tanner in 1981 developed and popularised LDPC codes, which became widely used in the 1990s. Low-hamming-weight code words and the simpler iterative decoding are the main attractions of LDPC codes. Parity bits are added to the transmitted bits in an LDPC encoder. The addition of the parity bits increases the number of bits in the codeword to be transmitted. These are corrupted by channel noise, which induces bit-flipping errors in the code words. The LDPC decoder uses these parity bits at the receiver to detect and correct errors (1,7,14). LDPC codes can achieve performance close to the channel capacity across a variety of channel models and have thus become one of the most popular families of forward error correction codes (8,11,21). They are used in digital communication standards, including Digital Video Broadcasting (e.g., DVB-S2X), WiMAX (IEEE 802.16e), and WiFi (IEEE 802.11n) (13,15). However, the main disadvantage of high error-correction efficiency is increased computational complexity in the decoder. Despite this, in recent years, many researchers have implemented LDPC decoders in software on many-core and multicore systems, for instance, in Software-Defined Radios (2,3). The performance of LDPC codes under message-passing algorithms is highly dependent on the structure of the code's distribution of short cycles in general and the Tanner graph, in particular. In the main substructure of the trapping sets, these cycles play a vital role in the error floor performance of LDPC codes (4). In the 5G mobile communication standard, polar codes are used for the control channels, and LDPC block codes (BC) are used for the data channels. These schemes still have obvious limitations despite their excellent performance. The BC-LDPC has several shortcomings, which include high complexity, long decoding latency, and slow convergence during decoding. Moreover, its performance is poor at small code lengths and low code rates. The polar code at long code lengths has unacceptably high decoding latency due to its serial decoding. In the future, 6G communication will require lower latency, higher reliability, and higher throughput to support real-time, high-rate data transmission. The convolutional code (CC) LDPC has lower decoding latency and complexity, and a lower error floor (5). Specified by a sparse parity-check matrix 'H', LDPC codes with row and column weights that are constant are said to be regular; otherwise, they are irregular. Regular codes do not perform as well as irregular ones. Irregular LDPC codes, unlike regular LDPC codes, can approach channel capacity when properly constructed. The comparison between the threshold signal-to-noise ratio (SNR) and the error-floor bit-error rate (BER) has been observed for both LDPC

codes and turbo codes. The column weight "wc" of the parity-check matrix determines the performance of LDPC codes. The number of ones in each column of the parity-check matrix determines the column weight of that column. Better BER performance is achieved with the large-column-weight codes. LDPC codes with wc = 2 are easier to implement than those with large column weights (wc>= 3)(6). The bipartite Tanner graph is a graph-based code in which the check nodes correspond to the parity-check constraints and the variable nodes correspond to the codeword symbols.The node degree plays an important role, where the connectivity among the nodes in the Tanner graph mainly characterises the error-correcting performance of a code.The node degree distribution in the Tanner graph can be specified in either the edge or the node perspective (9). It is not easy to design and construct an LDPC code with good performance if all of these structural properties are considered together (10). QC-LDPC have been adopted in several standards primarily for practical applications. A popular class of QC-LDPC codes is the codes developed using circulant permutation matrices (CPM). Regular QC-LDPC codes with an m × n array of N × N CPMs have been the subject of extensive research (11).

However, the high decoding complexity has been used to achieve performance gains, making it a challenging task to implement in practice (12,16). The QC-LDPC has good structural properties that yield capacity-approaching BER performance, facilitating hardware implementation for both encoding and decoding. However, only a limited set of fixed code rates can be achieved due to the regular QC-LDPC code-based rate-adaptive scheme. The small array size of the regular QC-LDPC code further increases the array size, thereby dramatically degrading BER performance. Therefore, it is more suitable to use irregular QC-LDPC codes (17). As observed in Ethernet, the lengths of frames and packets vary across many communication systems; thus, designing variable-length block LDPC codes is challenging. A desirable solution for transmitting variable-length packets, fortunately, is the use of convolutional LDPC codes, which allow the encoding and decoding of symbol sequences of arbitrary length (18). Various product codes have been proposed using Reed-Solomon (RS) and LDPC codes, such as RS-RS, RS-LDPC, and LDPC-LDPC, to correct long burst errors. A product code is composed of two block codes C1 (n1, k1) and C2 (n2, k2). Information bits are set in an array of k2 by k1. The rows and columns are encoded using C1 (inner code) and C2 (outer code), respectively(19). With an affinity for partial LDPC coding and Grey mapping, a high-order circular QAM constellation provides better BER performance than rectangular or cross QAM under time-varying phase noise (20).

Like binary LDPC codes, the construction methods for nonbinary LDPC codes can be divided into two major classes: random-like constructions by means of computer search and algebraic constructions based on finite fields, finite geometries, graph theory, and combinatorial designs. Algebraic LDPC codes have the advantages of easy hardware implementation for encoding and decoding, fast convergence in iterative decoding, and a low error floor. Furthermore, it is easy to construct algebraic LDPC codes with large minimum distances, which is extremely important for achieving good error performance (especially in the error-floor region). Moreover, research results show that the performance of an LDPC code when decoded with iterative algorithms can be improved efficiently as the row redundancies of its parity-check matrix increase. Generally, nonbinary LDPC codes with large row redundancies are constructed based on finite fields, finite geometries, and dispersed Reed-Solomon codes. But the field order of these codes will be fixed after the base matrices used in the code construction are determined. That is, the field-order transformation is not flexible (21). Low-density parity-check (LDPC) codes are widely adopted in digital communication standards today due to their powerful error-correction capabilities. The implementation of LDPC decoders on field-programmable gate arrays (FPGAs) typically begins with elaborating a register-transfer-level (RTL) description of a digital circuit to perform the computation. Producing such an RTL description, however, is a laborious task that requires knowledge of the hardware (22). This paper addresses an SDR system with real-time QC-LDPC Coding and decoding on an FPGA. It is organised as follows: Section II describes the SDR QAM4 system architecture. Section III describes the QC-LDPC Coder-Decoder system used in this paper in detail. Section IV describes the HDL Workflow Advisor for code generation. Section V illustrates the Simulation and Real-time results. Section VI concludes.

The SDR QAM4 SYSTEM Architecture: The top-level of the SDR QAM4 system architecture is illustrated in Figure 2. The PN sequence Generator generates the binary random data. It is a pseudo-noise sequence generator that generates Boolean noise in response to a polynomial. The PN sequence generator is illustrated in Figure 1. This Figure shows that the QAM-4 subsystem includes the complete transceiver, which runs on the FPGA in the FPGA-in-the-loop configuration. The vector scope shows the transmitted and received binary waveforms, which can be compared to detect any bit errors. The Error Rate Calculation Block calculates transmission errors. The AWGN Block is added, which implements the AWGN channel. The SNR of the circuit is varied in this block, and the transmission errors are calculated. The AWGN Block parameters are illustrated in Figure 4.

From Figure 3, the inserted bitstream from the PN Sequence Generator is passed through the S/P converter, where the incoming bitstream is converted into a parallel bitstream of 8bits. The 8-bit stream is passed into the LDPC Coder, which serves as the message vector. This is then multiplied by the Generator Matrix described in Section III to produce the 16-bit code word. The LDPC Coder is illustrated in Figure 5. From this figure, it can be seen that the Generator Matrix is in the Constant Block "G", which is an 8x16 matrix. This matrix is multiplied by the incoming message vector to produce the codeword. The matrix "G" construction is illustrated in the next section. The resulting 16-bit Vector is then passed through a P/S converter, which gives out a 2-bit stream that serves as the QAM4 symbol. This is then passed through a Demultiplexer block to separate the in-phase and the quad-phase bits, which are fed to the Modulator Subsystem. As shown in Figure 3, the sine and cosine signals for the modulation are generated automatically by the Numerically Controlled Oscillator (NCO) block, as illustrated in Figure 6. This signal is then multiplied by the Multiplexed signal to produce the modulated sine/cosine waves. The Modulator is illustrated in Figure 7. The QAM4 Multiplexer is illustrated in Figure 8. The QAM4 Multiplexer converts the incoming bitstream of "0" and "1" into polar values of "-1" and "1", respectively. All the Blocks and subsystems upto the Modulator Subsystem arein the transmitter; thereafter, all the Blocks and subsystems represent the receiver.

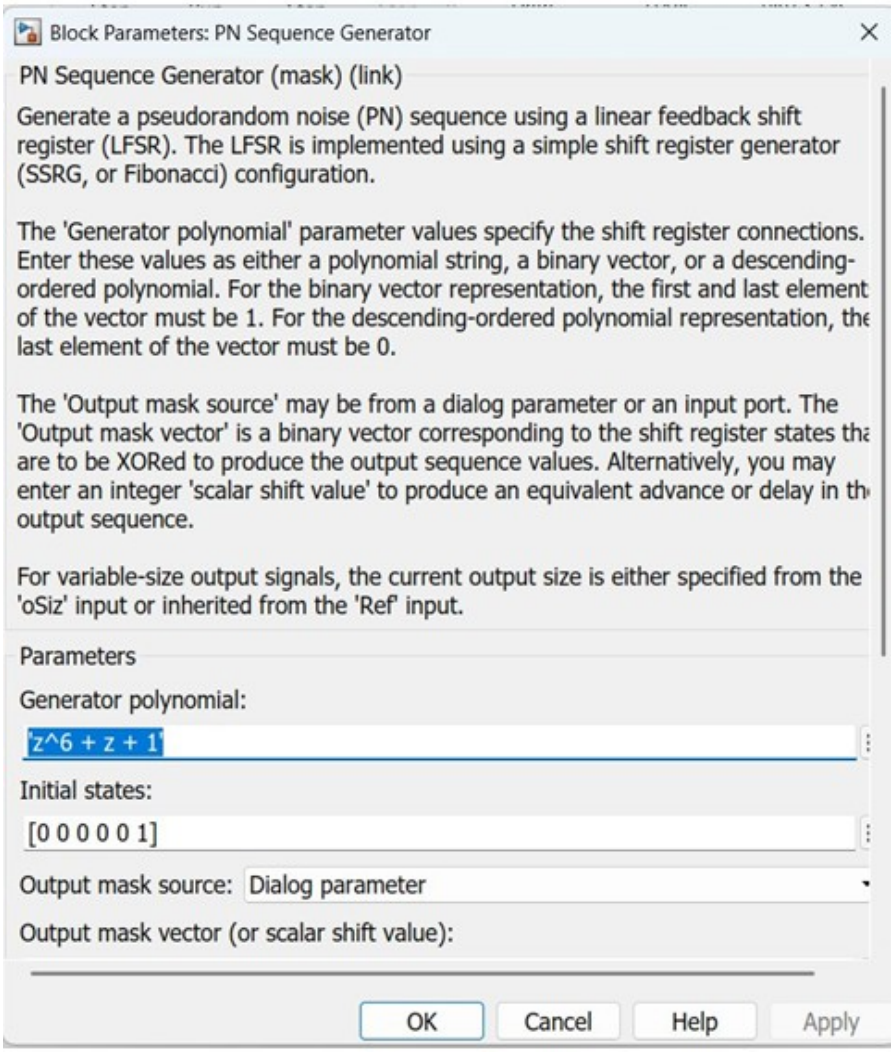

Fig. 1. PN Sequence Generator Block Details

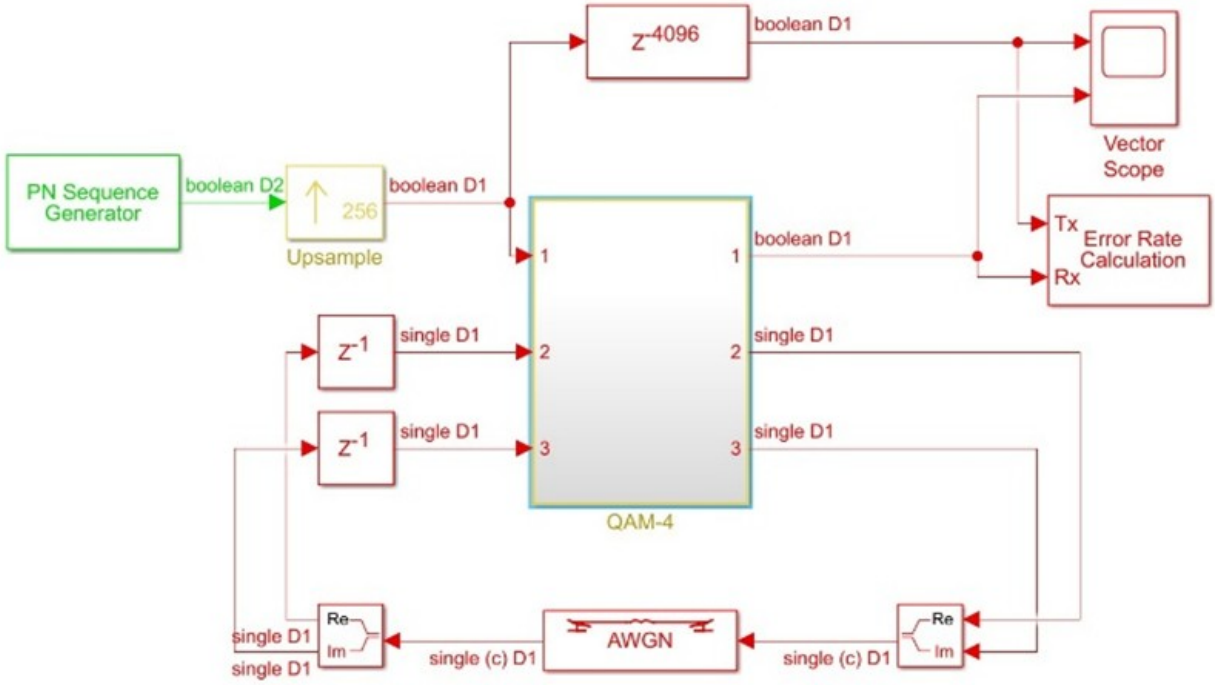

Fig. 2. The Top-Level Model of the SDR QAM-4 system

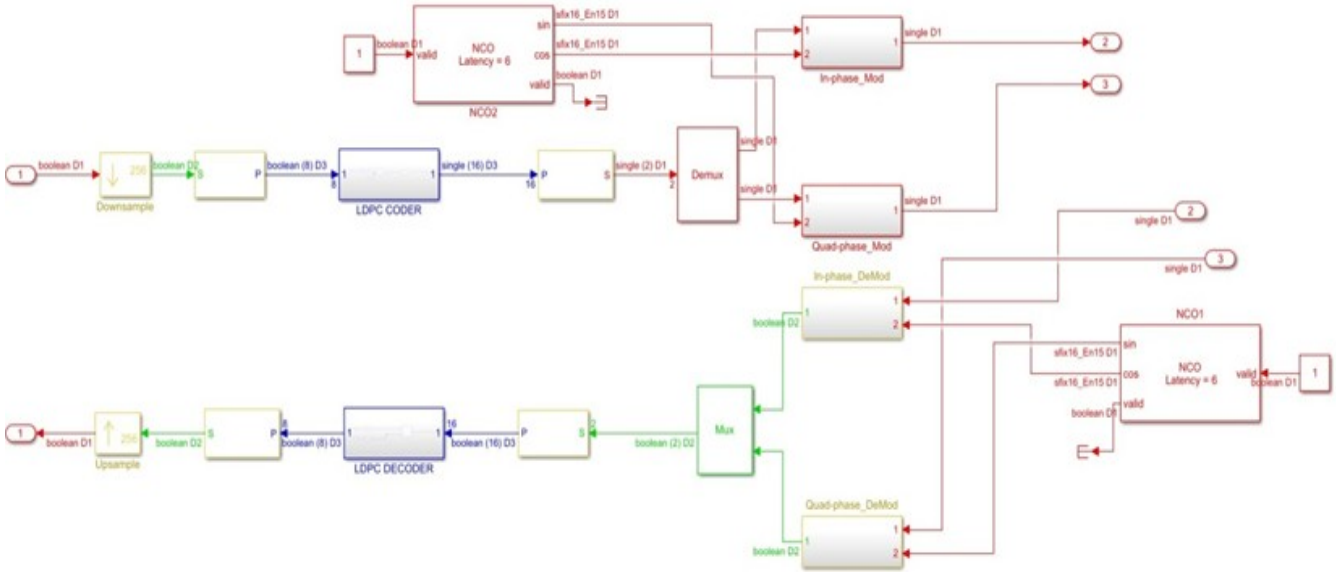

Fig. 3. The QAM-4 System Architecture, which runs on the FPGA

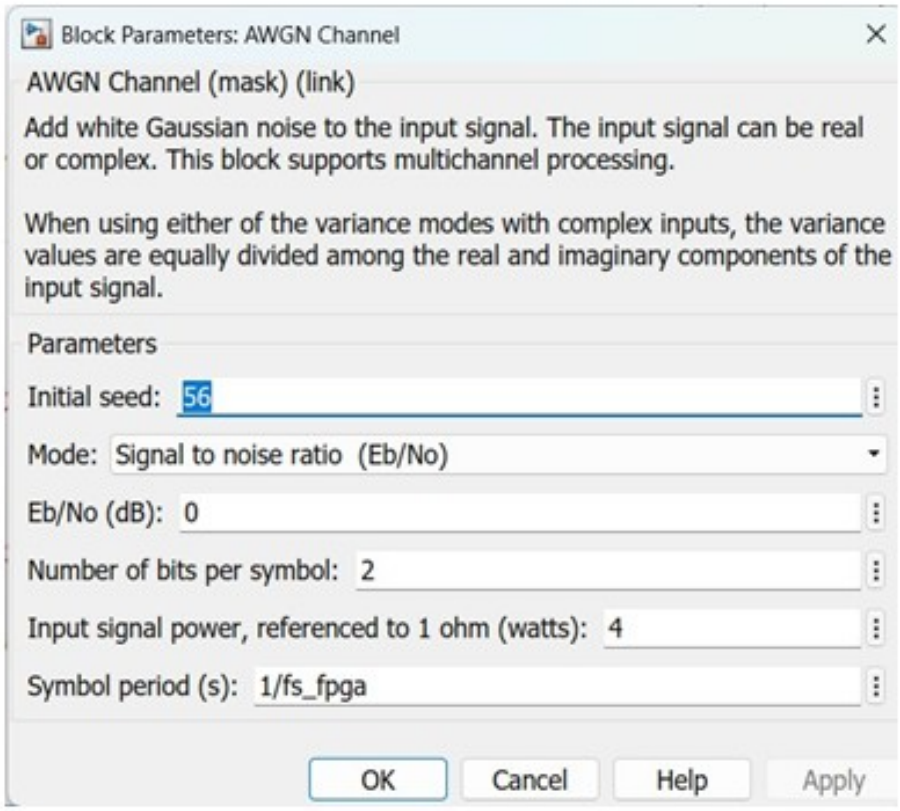

Fig. 4. The AWGN Block parameters

The modulated signal is then passed through the AWGN channel block, as seen in the top model of the system in Figure 2, which adds AWGN at the receiver's front end. This is then passed through the Demodulator subsystems for demodulation and eventually detection. Demodulation is performed by the multiplier block, followed by an FIR LPF, which filters out the high-frequency component and passes the DC component, which carries the information about the transmitted bit. The detector is the Threshold detector. The Demodulator is illustrated in Figure 9, and the detector is illustrated in Figure 10. The detector bits are then passed through the Multiplexer Block, which concatenates the in-phase and quad-phase bits into a 2-bit stream. This is then passed through the S/P converter to generate a 16-bit stream, which serves as the LDPC decoder's input. The LDPC Decoder is illustrated in Figure 11. Here, the 16 bits are used, and the 8-bit Syndrome is calculated. If the Syndrome is zero, no action is performed. But if the Syndrome is non-zero, action is performed in the form of bit flipping. The algorithm for this is described in Section III. After this block, the 8-bit data, which is the received bit stream, is passed through the P/S converter to convert it into a single-bit stream, which is then inserted into the Vector scope and the Bit Error Calculator for the final result. The results are illustrated in Section VII.

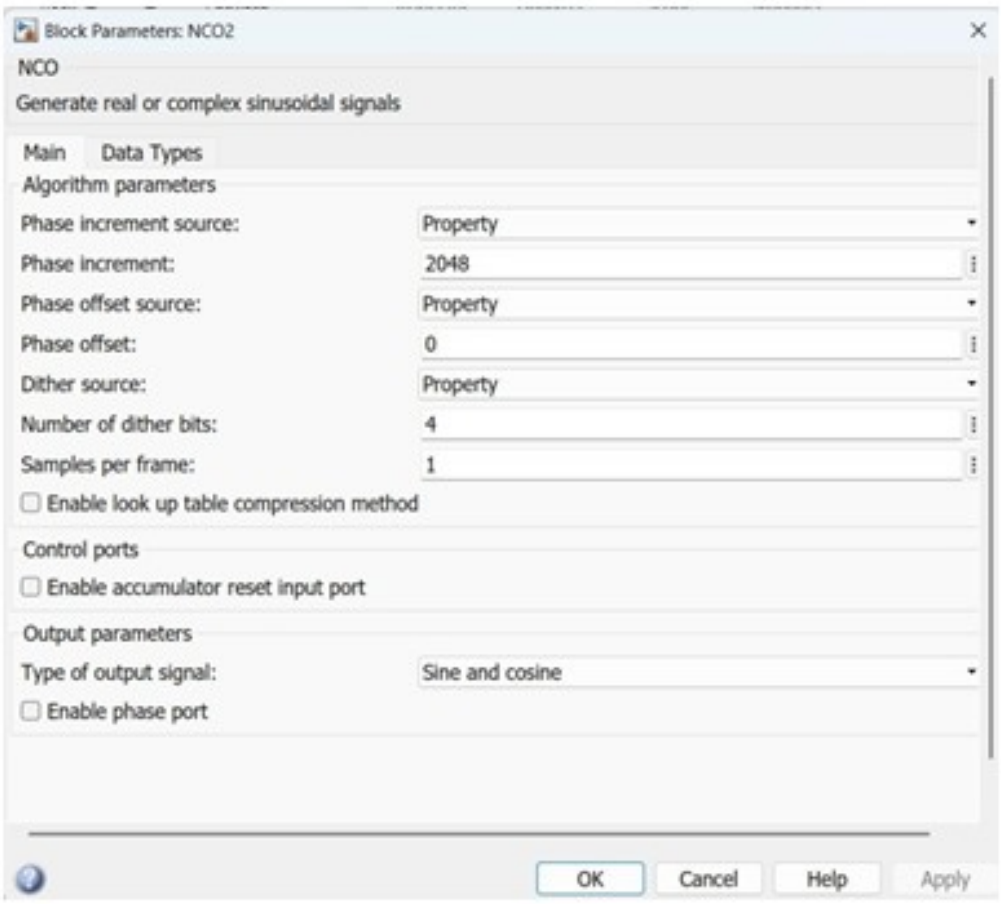

Fig. 6. The NCO Block Parameters

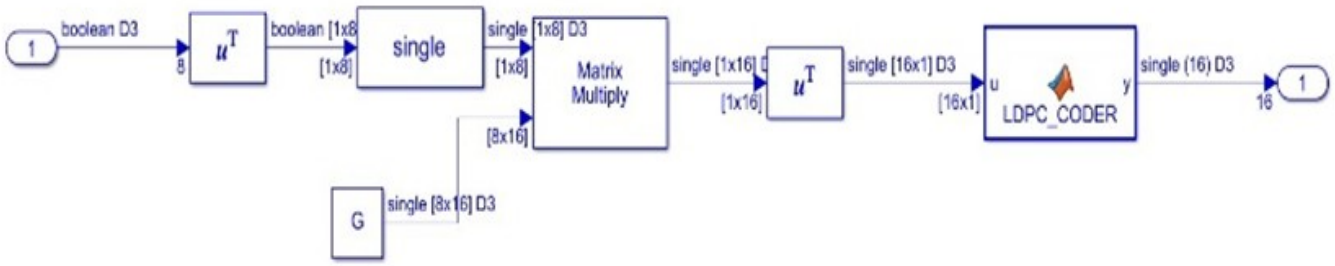

Fig. 5. The LDPC Coder Subsystem

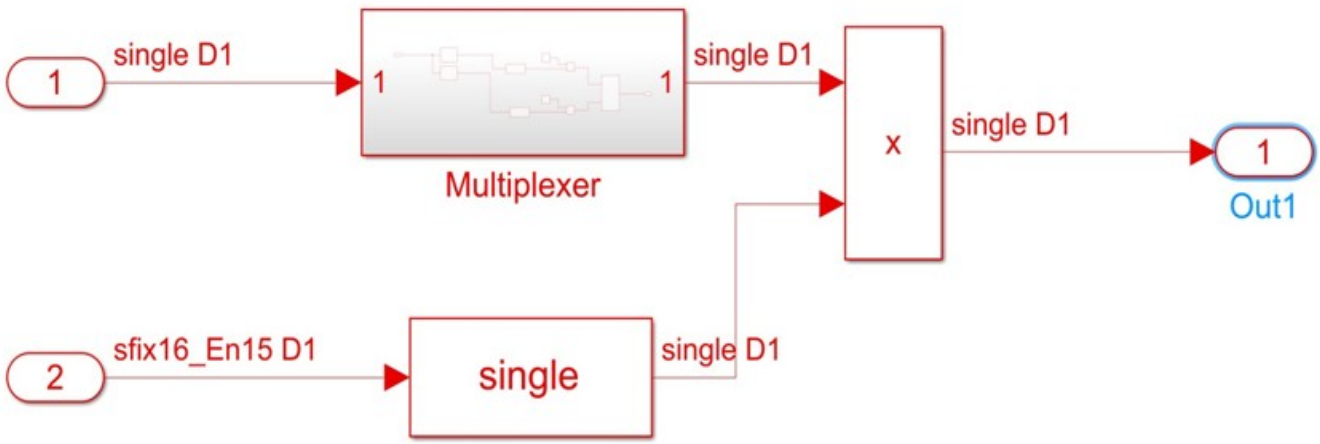

Fig. 7. The Sine / Cosine Modulator Subsystem for QAM4

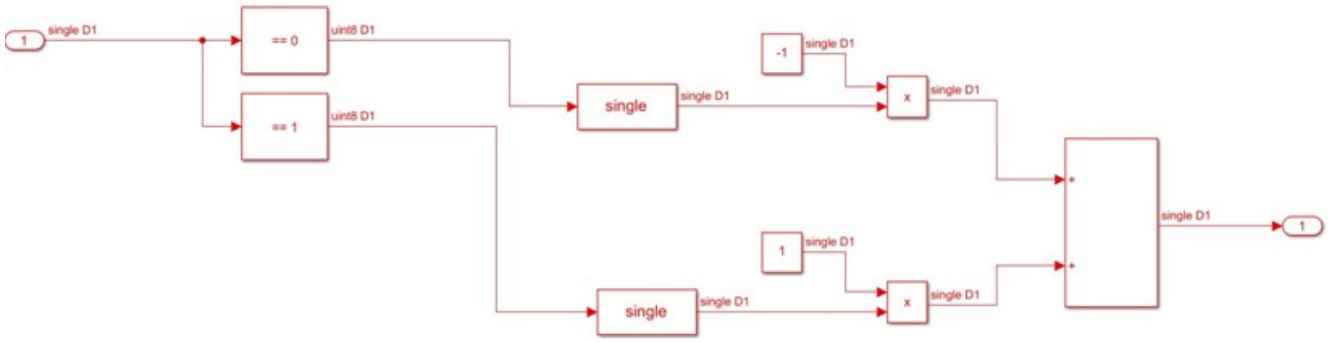

Fig. 8. The QAM4 Multiplexer Subsystem

Fig. 9. The In-phase and Quad-phase Demodulator/Detector Subsystem

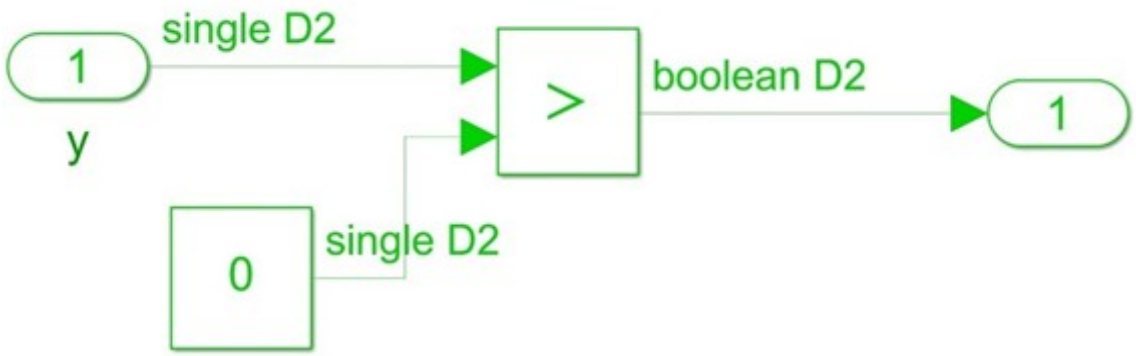

Fig. 10. The Threshold Detector Subsystem

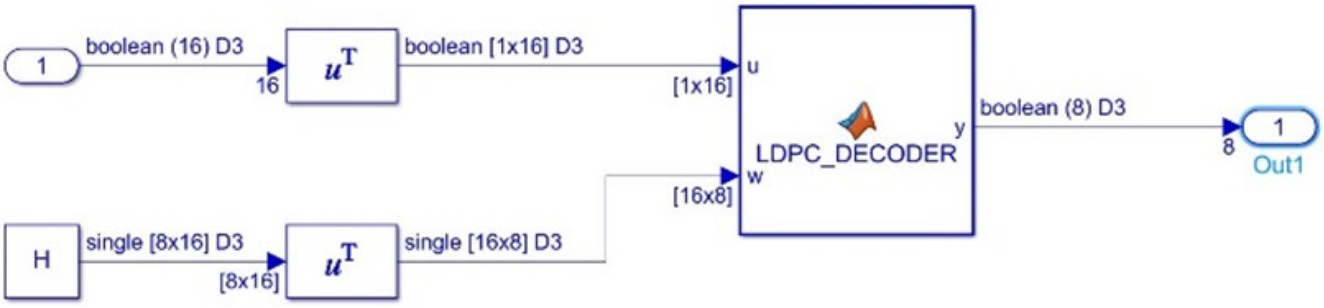

Fig. 11. The LDPC Decoder Subsystem

The QC-LDPC Coder and Decoder Algorithm

The H matrix, or the parity check matrix, is calculated as:

- QC-LDPC uses circulant blocks (shifted identity matrices). This structure makes hardware implementation much simpler and more efficient.
- Encoding Process: The QC-LDPC encoder takes input data bits and generates parity bits using the structured matrix, producing a longer encoded sequence that is resilient to noise.

The base matrix, or H matrix, or parity-check matrix is calculated from these shifted identity matrices. Here, the expansion factor is fixed at 4. So that the shifted identity matrices can take the following values: -1, 0, 1, 2, 3, each of these matrices is 4x4. The matrices for these values are given below:

$$h_{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

$$h_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$h_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

$$h_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (4)$$

$$h_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5)$$

The expansion or the base matrix H is given as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h_2} & \boldsymbol{h_3} & \boldsymbol{h_0} & \boldsymbol{h_{-1}} \\ \boldsymbol{h_1} & \boldsymbol{h_0} & \boldsymbol{h_{-1}} & \boldsymbol{h_0} \end{bmatrix} \quad (6)$$

Expanding each term, we get

$$H = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

This matrix is of the form: $\boldsymbol{H} = [\boldsymbol{P} \quad \boldsymbol{I_K}]$

Where K = 8 and $\mathbf{I_K}$ is an 8 x 8 identity matrix;

Hence, the Generator matrix from this matrix is defined as $\boldsymbol{G} = [\boldsymbol{I_K} \quad \boldsymbol{P^I}]$.

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

Both the **H** and **G** matrices are sparse. The number of ones is much lower than the number of zeros. The coding rate (CR) is given from the H matrix as

$$CR = 8/16 \tag{9}$$

Hence,

$$CR = 1/2 \tag{10}$$

The transmitted code is calculated as

$$\boldsymbol{X = m \times G} \tag{11}$$

Where **m** is the message vector, and **G** is the Generator Matrix. We observe that the first 8 bits in this output codeword are the message vector and the next 8 bits constitute the parity bits. This code is transmitted, and at the receiver, the corrupted code is obtained after passing through AWGN.

$$\boldsymbol{Y = X + E} \tag{12}$$

Where Y is the corrupted code word (16-bit Vector), and E is the Error vector added (16-bit Vector). The addition is modulo 2. From this corrupted code word, the Syndrome is calculated as

$$\boldsymbol{S = Y \times H^T} \tag{13}$$

From S, 8 equations can be derived as the components of the S vector:

$$S_1 : Y_3 + Y_8 + Y_9 \tag{14}$$

$$S_2 : Y_4 + Y_5 + Y_{10} \tag{15}$$

$$S_3 : Y_1 + Y_6 + Y_{11} \tag{16}$$

$$S_4 : Y_2 + Y_7 + Y_{12} \tag{17}$$

$$S_5 : Y_2 + Y_5 + Y_{13} \tag{18}$$

$$S_6 : Y_3 + Y_6 + Y_{14} \tag{19}$$

$$S_7 : Y_4 + Y_7 + Y_{15} \tag{20}$$

$$S_8 : Y_1 + Y_8 + Y_{16} \tag{21}$$

For no error, all these components of the S vector are zero. If any component is 1, the bit-flipping algorithm is applied to set it to 0. Let us take an example of S_1 being non-zero. The following algorithm is performed to make it zero:

```
S = Y * H^T

if(S(1)==1)
      if(Y(3) == 0)
Y(3) = 1;
      else
Y(3) = 0;
end
end

S = Y * H^T;
```

```
        if(S(1)==1)
            if(Y(3) == 0)
Y(3) = 1;
            else
Y(3) = 0;
            end


            if(Y(8)==0)
Y(8)=1;
            else
Y(8)=0;
            end
        end
```

$S = Y * H^T;$

```
        if(S(1)==1)
            if(Y(8)==0)
Y(8)=1;
            else
Y(8)=0;
            end


            if(Y(9)==0)
Y(9)=1;
            else
Y(9)=0;
            end
        end
```

This shows that from Equation (14) $S_1 == 1$ only when the 3rd, 8th, or 9th bit of the Y vector is bit flipped. Hence, we first flip the 3rd bit of Y and check for Syndrome. If it is not zero, we revert back the 3rd bit and flip the 8th bit of Y. Again, check for the Syndrome. If it is still 1, then we revert the 8th bit of Y and flip the 9th bit. We observe that Syndrome S_1 is now 0. Here, there is one thing to notice: both S_1 and S_6 have one component in common, and that is Y_3. When it is in error, after flipping it, the S_6 automatically corrects itself. Likewise, S_1 and S_8 have Y_8 in common, and if we flip it, both these components are taken care of; that is, S_1 and S_8 become 0. This process is repeated for S_2-S_8, and the Syndrome is checked each time. We observe that after we operate on S_8, the Syndrome comes out to be zero. This process removes the errors even if all the 8 bits are in error. That is, almost 100% error removal. The BER vs SNR curve is shown in Section V.

HDL Workflow Advisor for Code Generation: The HDL workflow advisor is invoked from MATLABby right-clicking the QAM4 Subsystem. This is shown in Figures 12 and 13.Figure 12 shows the parameters of the Target Device and the Synthesis Tool used to synthesise the circuit. Figure 13 shows the parameters of the RTL Code and the Testbench. The Target device, as seen in Figure 12, is the Xilinx Zynq Ultrascale+ RFSoC ZCU111 Evaluation Kit, and the Synthesis Tool chosen is Xilinx Vivado version 2020.2. Figure 13 shows that the VHDL and C++ codes were generated along with the test bench. Each block in the HDL Subsystem is generated in a separate VHDL File. All processes in the HDL Workflow Advisor should succeed in generating the ".bit" and ".exe" files for the FPGA and the ARM Cortex processors, respectively. In the last step of the chain, the FPGA-in-the-Loop Model is generated. This is the model that actually runs the ".bit" and ".exe" files on the FPGA and ARM processor. This model is illustrated in Figure 14. Figure 15 illustrates the FPGA-in-the-Loop dialogue box for loading the ".bit" file onto the FPGA board. From the above Figure, it can be seen that as we press the "Load" button, the QAM_4_fil.The bit file is loaded onto the FPGA board. After loading, we return to the FIL Model generated by the HDL Workflow Advisor and press the "Play" button on the top ribbon. As soon as we press this button, the model starts running in real time, and we get real-time results on the Vector Scope. Moreover, the Error Vector Calculator computes transmission errors, and we can plot the BER vs. SNR curve of the circuit under test by recording the BER as we repeatedly vary the SNR.

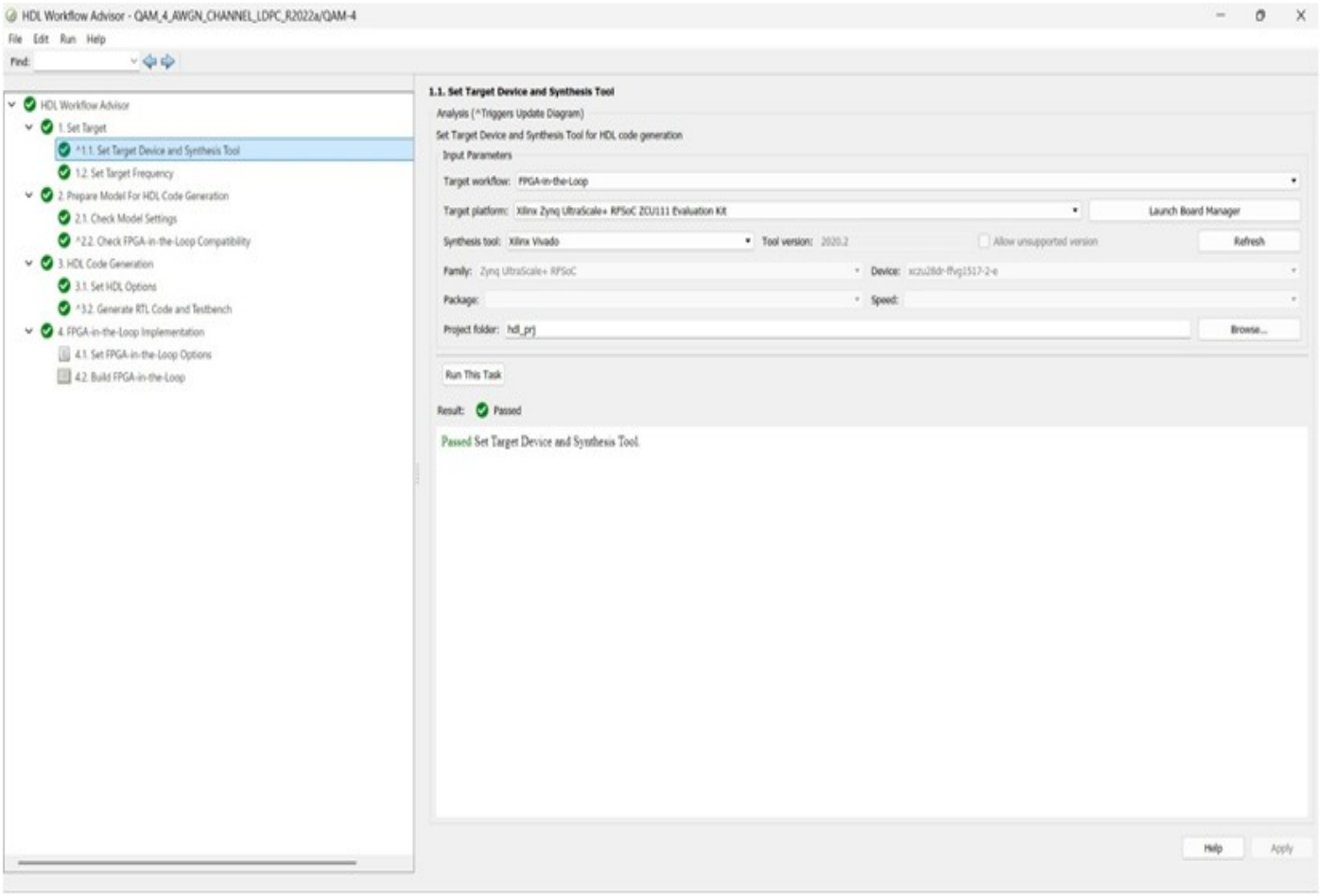

Fig. 12. HDL Workflow Advisor – Set Target Device and Synthesis Tool

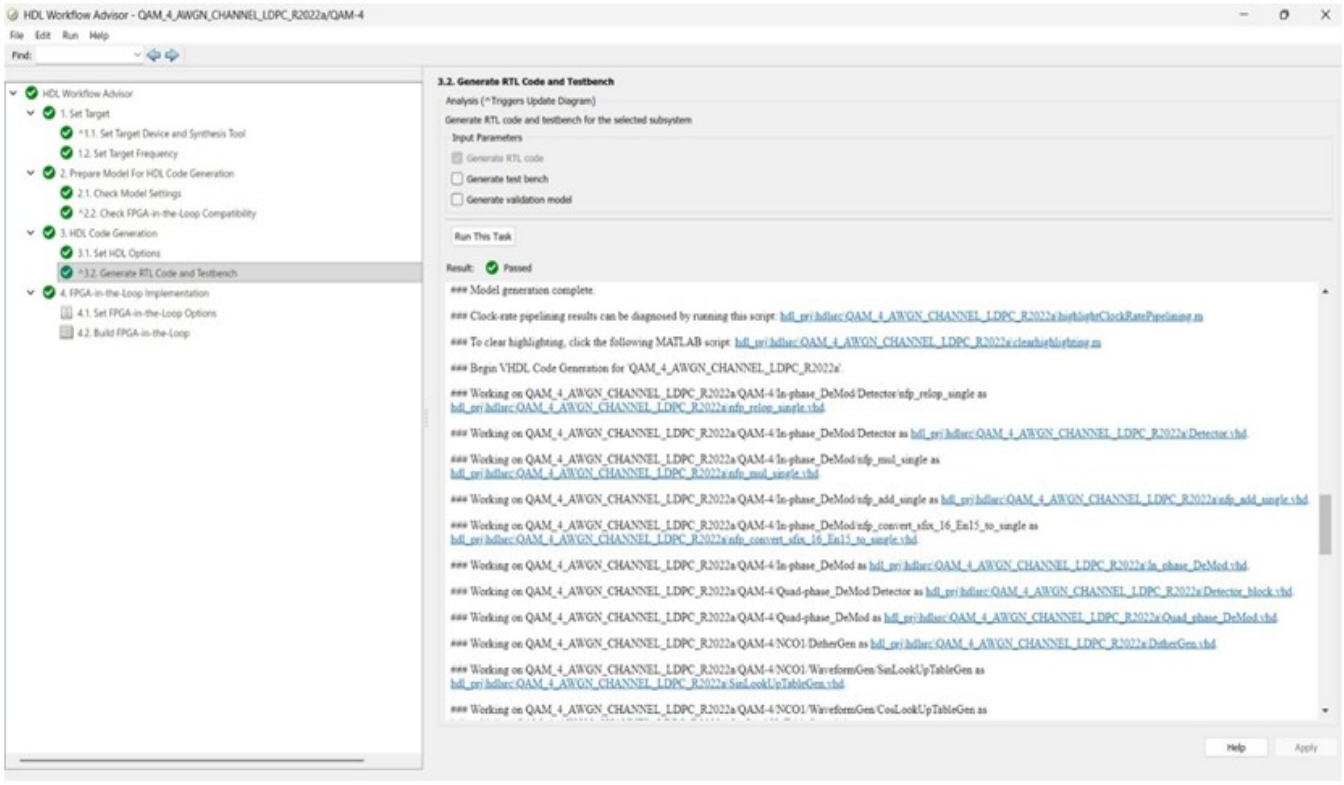

Fig. 13. HDL Workflow Advisor –Generate RTL Code and Testbench

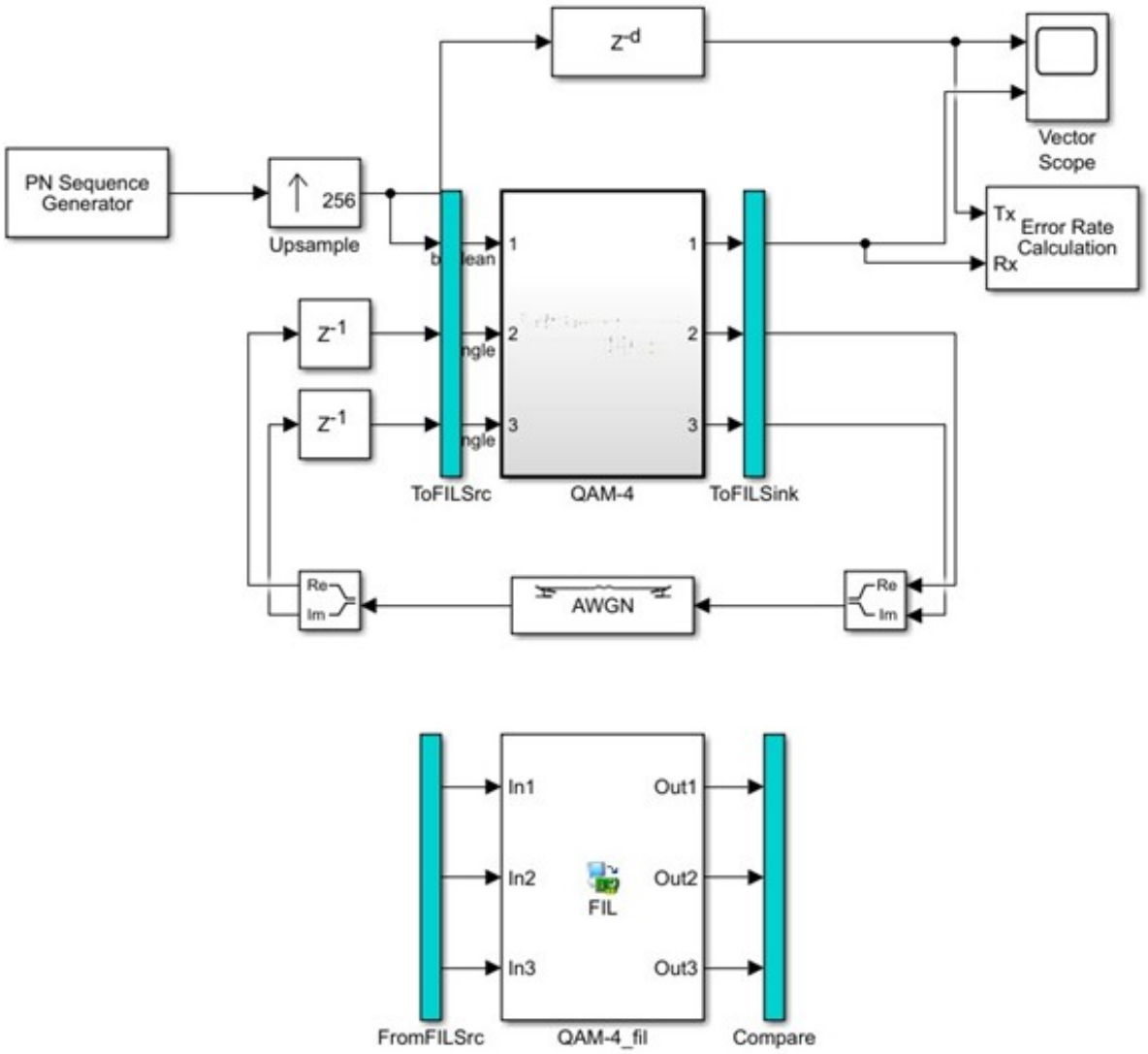

Fig. 14. The FPGA-in-the-Loop Model Generated automatically by HDL Workflow Advisor

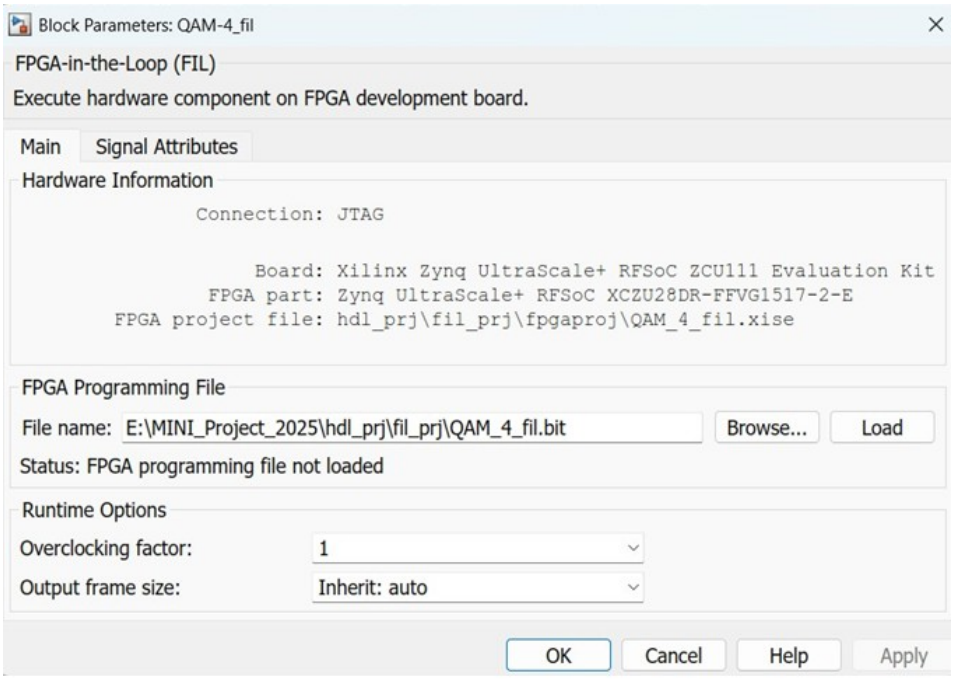

Fig. 15. The QAM-4 Fil Dialogue Box for Loading the ".bit" File on the FPGA board

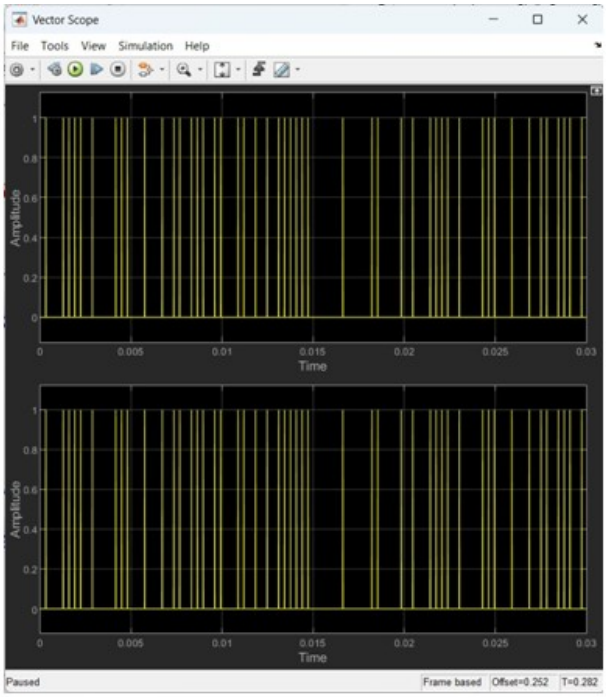

Fig. 16. The Simulation Results for SNR=10

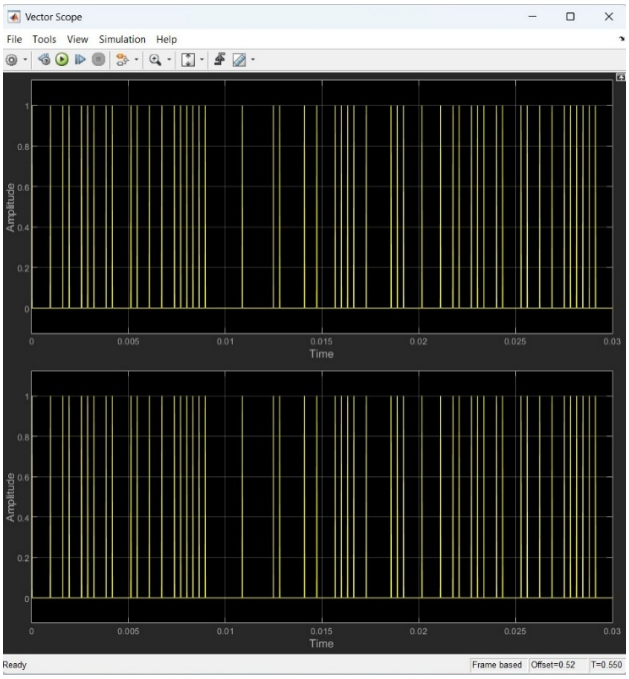

Fig. 17. The real-time results for SNR = 10

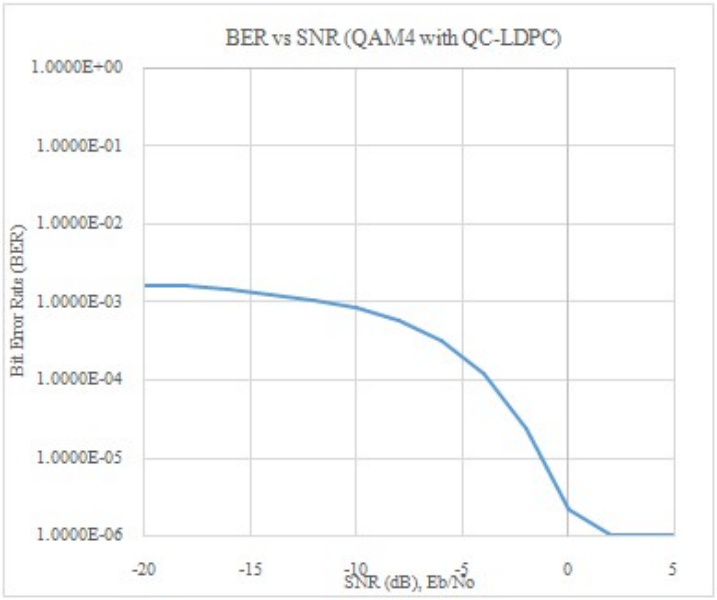

Fig. 18. BER vs SNR (Eb/No) Curve for QAM4, QC-LDPC

Simulation and Real-Time Results: Figures 16 and 17 illustrate the Simulation and real-time results for this model, respectively. This is performed for the SNR=10. From these results, it can be observed that the transmitted bitstream is the replica of the received bitstream. The yellow bars represent a 1, and the absence of a bar represents 0. The BER vs SNR curve for this model is illustrated in Figure 18. After SNR = 0, there are no errors.

CONCLUSIONS

The SDR QAM4 transceiver is designed and tested in real-time in MATLAB/Simulink. The QC-LDPC Coder and Decoder are implemented on the FPGA and are working well. The channel used in the model is an AWGN channel. The circuit's noise performance shows that it works well even at negative SNR values.The real-time and Simulation results are identical. This indicates that the overall circuit and the SDR QAM4 system work fine in Simulation and in real time. The Model-based design, thus realised, saves a tremendous amount of time in design, coding, and real-time testing.

REFERENCES

Felipe Costa Pais, Ricardo Seriacopi Rabaça, Vinicius Piro Barragam, Fadi Jerji, and Cristiano Akamine, "Evaluation of LDPC codes and Layered Division Multiplexing in Digital Radio Mondiale Plus," 2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 04-06 August 2021, DOI: 10.1109/BMSB53066.2021.9547138, Chengdu, China.

Bertrand Le Gal and Christophe Jego, "High-Throughput Multicore LDPC Decoders Based on x86 Processor," IEEE Transactions on Parallel and Distributed Systems (Volume: 27, Issue: 5, 01 May 2016), pp. 1373 – 1386, DOI: 10.1109/TPDS.2015.2435787.

Rongchun Li, Jie Zhou, Yong Dou, Song Guo, Dan Zou, and Shi Wang, "A multi-standard efficient column-layered LDPC decoder for Software Defined Radio on GPUs," 2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 16-19 June 2013, DOI: 10.1109/SPAWC.2013.6612145, Darmstadt, Germany.

Ali Dehghan and Amir H. Banihashemi, "On the Tanner Graph Cycle Distribution of Random LDPC, Random Protograph-Based LDPC, and Random Quasi-Cyclic LDPC Code Ensembles," IEEE Transactions on Information Theory (Volume: 64, Issue: 6, June 2018), pp. 4438 – 4451, DOI: 10.1109/TIT.2018.2805906.

Kun Zhu and Zhanji Wu, "Comprehensive Study on CC-LDPC, BC-LDPC and Polar Code," 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 06-09 April 2020, DOI: 10.1109/WCNCW48565.2020.9124897, Seoul, Korea (South).

Vibha Kulkarni and K. Jaya Sankar, "Design of structured irregular LDPC codes from structured regular LDPC codes," Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), 07-08 February 2015, DOI: 10.1109/C3IT.2015.7060128, Hooghly, India.

Xiao Ma,Qianfan Wang, Mangang Xie, and Suihua Cai, "Implicit Globally-Coupled LDPC Codes Using Free-Ride Coding," 2022 IEEE Wireless Communications and Networking Conference (WCNC), 10-13 April 2022, DOI: 10.1109/WCNC51071.2022.9771931, Austin, TX, USA.

Yang Yu, Ziyan Jia, Weige Tao, and Shiliang Dong, "LDPC codes optimisation for differential encoded LDPC coded systems with multiple symbol differential detection," 2016 IEEE 5th Global Conference on Consumer Electronics, 11-14 October 2016, DOI: 10.1109/GCCE.2016.7800541, Kyoto, Japan.

Sachini Jayasooriya, Mahyar Shirvanimoghaddam, Lawrence Ong, Gottfried Lechner, and Sarah J. Johnson, "A New Density Evolution Approximation for LDPC and Multi-Edge Type LDPC Codes," IEEE Transactions on Communications (Volume: 64, Issue: 10, October 2016), pp. 4044 – 4056, DOI: 10.1109/TCOMM.2016.2600660.

Hengzhou Xu, Hai Zhu, Mengmeng Xu, Bo Zhang, and Sifeng Zhu, "Girth Analysis of Tanner (5,11) Quasi-Cyclic LDPC Codes," 2018 14th International Conference on Computational Intelligence and Security (CIS), 16-19 November 2018, DOI: 10.1109/CIS2018.2018.00053, Hangzhou, China.

Alireza Tasdighi, Amir H. Banihashemi, and Mohammad-Reza Sadeghi, "Symmetrical Constructions for Regular Girth-8 QC-LDPC Codes," IEEE Transactions on Communications (Volume: 65, Issue: 1, January 2017), pp. 14 – 22, DOI: 10.1109/TCOMM.2016.2617335.

Xiangcheng Li, Tuanfa Qin, Haiqiang Chen, Youming Sun, Qi Liang, and Liping Luo, "Hard-Information Bit-Reliability Based Decoding Algorithm for Majority-Logic Decodable Nonbinary LDPC Codes," IEEE Communications Letters (Volume: 20, Issue: 5, May 2016), pp. 866 – 869, DOI: 10.1109/LCOMM.2016.2537812.

Saleh Usman, Mohammad M. Mansour, and Ali Chehab, "A Multi-Gbps Fully Pipelined Layered Decoder for IEEE 802.11n/ac/ax LDPC Codes," 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 03-05 July 2017, DOI: 10.1109/ISVLSI.2017.42, Bochum, Germany.

Hengzhou Xu and Baoming Bai, "Superposition Construction of Q -Ary LDPC Codes by Jointly Optimising Girth and Number of Shortest Cycles," IEEE Communications Letters (Volume: 20, Issue: 7, July 2016), pp. 1285 – 1288, DOI: 10.1109/LCOMM.2016.2564401.

Ahmed Mahdi, Nikos Kanistras, and Vassilis Paliouras, "A Multirate Fully Parallel LDPC Encoder for the IEEE 802.11n/ac/ax QC-LDPC Codes Based on Reduced Complexity XOR Trees," IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume: 29, Issue: 1, January 2021), pp. 51 – 64, DOI: 10.1109/TVLSI.2020.3034046.

Liqian Wang, Dongdong Wang, Yongjing Ni, Xue Chen, Midou Cui, and Fu Yang, "Design of irregular QC-LDPC code based multi-level coded modulation scheme for high speed optical communication systems," China Communications (Volume: 16, Issue: 5, May 2019), pp. 106 – 120, DOI: 10.23919/j.cc.2019.05.009.

Liwei Mu, "Ensemble of high performance structured binary convolutional LDPC codes with moderate rates," China Communications (Volume: 17, Issue: 10, October 2020), pp. 195 – 205, DOI: 10.23919/JCC.2020.10.014.

Seongkwon Jeong and Jaejin Lee, "Iterative LDPC–LDPC Product Code for Bit Patterned Media," IEEE Transactions on Magnetics (Volume: 53, Issue: 3, March 2017), Article Sequence Number: 3100704, DOI: 10.1109/TMAG.2016.2618008.

Bin Zheng, Lianjun Deng, Mamoru Sawahashi, and Norifumi Kamiya, "High-order circular QAM constellation with high LDPC coding rate for phase noise channels," 2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC), 17-20 December 2017, DOI: 10.1109/WPMC.2017.8301807, Bali, Indonesia.

Nishil Talati, Zhiying Wang, and Shahar Kvatinsky, "Rate-compatible and high-throughput architecture designs for encoding LDPC codes," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 28-31 May 2017, DOI: 10.1109/ISCAS.2017.8050836, Baltimore, MD, USA.

Hengzhou Xu, Dan Fen, Cheng Sun, and Baoming Bai, "Algebraic-based nonbinary ldpc codes with flexible field orders and code rates," China Communications (Volume: 14, Issue: 4, April 2017), pp. 111 – 119, DOI: 10.1109/CC.2017.7927569.

Bingbing Wang, Jing Kang, and Yan Zhu, “Performance Balanced General Decoder Design for QC-LDPC Codes Using Vivado HLS,” 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), 18-20 June 2021, DOI: 10.1109/ICEIEC51955.2021.9463813, Beijing, China.

****