mathematics

MDPI

Article

FPGA-Oriented LDPC Decoder for Cyber-Physical Systems

Mateusz Kuc *, Wojciech Sułek and Dariusz Kania

Institute of Electronics, Silesian University of Technology, ul. Akademicka 2A, 44-100 Gliwice, Poland; wojciech.sulek@polsl.pl (W.S.); dariusz.kania@polsl.pl (D.K.)

* Correspondence: mateusz.kuc@polsl.pl

Received: 7 April 2020; Accepted: 29 April 2020; Published: 4 May 2020

Abstract: A potentially useful Cyber-Physical Systems element is a modern forward error correction (FEC) coding system, utilizing a code selected from the broad class of Low-Density Parity-Check (LDPC) codes. In this paper, development of a hardware implementation in an FPGAs of the decoder for Quasi-Cyclic (QC-LDPC) subclass of codes is presented. The decoder can be configured to support the typical decoding algorithms: Min-Sum or Normalized Min-Sum (NMS). A novel method of normalization in the NMS algorithm is proposed, one that utilizes combinational logic instead of arithmetic units. A comparison of decoders with different bit-lengths of data (beliefs that are messages propagated between computing units) is also provided. The presented decoder has been implemented with a distributed control system. Experimental studies were conducted using the Intel Cyclone V FPGA module, which is a part of the developed testing environment for LDPC coding systems.

Keywords: cyber physical systems; LDPC; QC-LDPC; FPGA; min-sum; normalized min-sum; distributed control system; token ring

1. Introduction

In recent years, there has been a strengthening link between advancement in computational technologies and components of physical systems. The so-called Cyber-Physical System (CPS) consists of a set of modules that interact with each other and communicate with the outside world. Combining computational and communication aspects with control techniques in one system becomes a challenge. Cyber-Physical Systems applications can be found in almost all areas of human life, such as production systems, intelligent networks, robotics, transport systems, medical devices, military systems, home networks, intelligent buildings, etc. In many CPS applications, digital communication systems play a key role, as they are often integrated with the executive system. During communication, data may be disturbed by various unwanted signals, noise and/or interferences. Error Correction Coding (ECC) techniques can not only detect communication errors, but also reconstruct valid data. Low-Density Parity-Check (LDPC) codes are one of the best known codes with very good correction capabilities.

LDPC codes were firstly presented in 1962 by R. G. Gallager [1], but the technology of that time did not allow practical applications. Research and development of LDPC codes was resumed in 1999 after an article by D. J. C. MacKay [2]. Today, LDPC codes have been already used in various applications, the DVB-S2 and DVB-T2 (digital television) standards [3], 10-Gbase-T Ethernet networks [4], ITU-T G.hn [5], 802.11ad (WiGig) [6], 802.11n/ac/ax (WiFi) and 802.16e (WiMAX) [7]. In 2015, the first information about applications of LDPC codes in 5G networks was revealed [8].

An LDPC code is defined by its parity check matrix, which is a sparse matrix, or equivalently, by the corresponding bipartite factor graph (known as a Tanner graph [9]). The graph structure has a direct relationship with the structure of the LDPC encoder and decoder. Particularly important for implementation reasons are the Quasi-Cyclic (QC) codes [10], for which parity check matrix is an

array of square submatrices, with every submatrix being a cyclic permutation of an identity matrix. These types of arrays allow for efficient hardware implementation of the parallel and semi-parallel QC-LDPC decoder [11].

The research activities in the area of LDPC coding techniques are still prevalent and they include, among others, the methodical construction of LDPC codes [10,12,13], decoding algorithm design [11,14,15] and efficient decoder implementation [4,16–18]. The implementation issues typically constraint the design to some class of implementation oriented LDPC codes, which most commonly belong to the mentioned QC-LDPC class.

LDPC codes are decoded iteratively using the sum-product algorithm, also known as belief propagation [2], which closely approximates maximum-likelihood decoding. The commonly used message passing schedule is a two-phase message passing scheme [11], where a decoding iteration is divided into two rounds of computations, corresponding to variable and check nodes of the code graph respectively. There are also known other schedules, for example the layered decoding scheme [19].

Every hardware LDPC decoder belongs to one of the categories: serial, semi-parallel or parallel, which means the decoder computation units execute the decoding algorithm in a serial, semi-parallel or parallel manner, respectively. The serial decoder has the lowest throughput, but also the lowest hardware utilization. The parallel decoder has the highest throughput and hardware requirements. A semi-parallel decoder is a compromise solution that is the best choice in most applications. The results of the research presented in this paper concern a semi-parallel decoder architecture with the two-phase message computation schedule. The presented decoder solution can be adapted to any QC-LDPC code. There exist a broad range of known implementations, most of them surveyed in [16], many of them aimed at ASIC implementation [16,18], some of them for implementation in software [17]. The research presented in this paper is devoted specifically to the implementation in a Field-Programmable Gate Array (FPGA) chip. In the designed decoder, an important parts of the computing units are fitted directly to the Look-Up-Table (LUT) fabric of FPGAs, giving a slightly improved decoding performance and decreased resources requirements.

An important features of the LDPC decoder is the implemented method of calculating messages in computing units, which is essentially independent of the chosen architecture. Computationally efficient message computation algorithms are known as Min-Sum (MS) and Normalized Min-Sum (NMS) [14,15]. The NMS algorithm differs from the Min-Sum algorithm by the additional normalization stage, which slightly reduces the magnitude of the iteratively approximated beliefs, which has a known effect of improved final decoding results.

The main scope of this paper is a presentation of an irregular QC-LDPC decoder implementation, which is oriented specifically to a LUT based structure of an FPGA programmable chip. The essence of the contribution of this paper is technology mapping approach, in which the normalization in the NMS algorithm is directly oriented to the LUT-based architecture. Moreover, we present the decoder design and resulting system solutions, aimed at an efficient implementation of the LDPC decoder inside a LUT-based FPGAs.

This paper is organized as follows. The second chapter presents the theoretical background of LDPC decoding and implementation of QC-LDPC decoder. Next, new concepts of technology mapping of QC-LDPC decoder oriented to FPGA are proposed. The implementation of distributed control unit of QC-LDPC decoder and the original implementation of normalization module which is oriented to LUT-based architecture are presented. Section 4 illustrates experimental results of proposed solutions. The obtained results were compared with solutions known from the literature. The article ends with a summary, which contains directions for future work.

2. Theoretical Background of Hardware Implementation of a Semi-Parallel QC-LDPC Decoder

An LDPC code is defined by a parity check matrix $\mathbf{H}$ of size $M \times N$. An example matrix $\mathbf{H}$ of a QC type code is shown in Figure 1 in the form of an array of its $M \times N = 8 \times 16$ entries. Matrix $\mathbf{H}$ is a QC matrix, since it consists of circulant submatrices of size $P \times P = 4 \times 4$. A submatrix will in short be

called the matrix **P**. Parity check matrix **H** of QC-LDPC code can be presented in the form of an array of submatrices, where "X" corresponds to the all-zero submatrix and a numerical value s corresponds to a an identity matrix circularly shifted by s positions to the right (i.e., columns are cyclically shifted by the indicated number). For QC-LDPC codes, storing the **H** structure in the decoder (RAM/ROM) memory requires only storing the array of cyclic shift values. The implementation of the presented decoder takes advantage of this simplified representation.

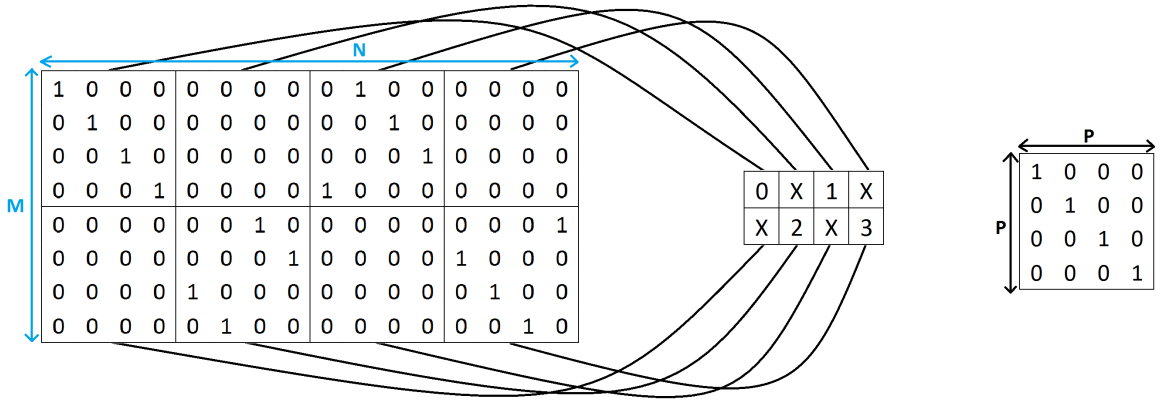

Figure 1. An example matrix **H** of the QC type code.

An example parity check matrix of QC-LDPC code, constructed for this research with a method presented in [20] , is shown in Figure 2. The size of the submatrix is $P = 64$ in this case, the size of the matrix **H** is 512×1024 and the code rate is $R = (N - M)/N = 1/2$. The minimum number of non-zero elements in rows is 6 and the maximum number is 7. For columns these are respectively 2 and 7. It means this is an irregular code with the maximum column weight of 7.

0	X	12	9	X	0	X	17	X	0	X	X	X	X	X	X
7	0	5	X	13	X	8	X	0	X	0	X	X	X	X	X
1	X	X	46	1	X	X	X	1	12	X	0	X	X	X	X
X	0	X	X	X	0	X	X	3	X	7	11	0	X	X	X
15	X	X	X	1	X	X	6	5	X	X	X	53	0	X	X
9	X	0	X	X	0	X	X	1	5	X	X	X	3	0	X
13	30	X	X	X	X	0	X	8	X	X	X	50	63	X	0
8	X	X	1	X	X	X	X	16	X	61	X	X	X	23	15

Figure 2. QC-LDPC offset matrix for $P = 64$ and $R = 1/2$.

LDPC decoding algorithms with complexity scaling as $O(N)$ belong to the class of iterative belief propagation (BP) algorithms, where beliefs are propagated between nodes represented as a Tanner graph. This graph is a bipartite graph with control vertices representing rows of **H**, bit vertices representing columns of **H** and edges representing non-zero positions in **H**. The commonly used method for determining the beliefs (messages propagated in the algorithm) are Min-Sum and NMS [15]. The decoding process utilizing Min-Sum function can be presented in consecutive steps as follows:

1. *Initialization*: Assigning input values that are Log-Likelihood Ratio (LLR). For every $m \in \mathcal{M}(n)$ and $n \in (1, N)$:
$$Q_{nm} := L(x_n|y_n) = \ln\left(\frac{P(x_n = 0|y_n)}{P(x_n = 1|y_n)}\right) \tag{1}$$

2. *Control nodes*: Determination of minimum values. For every $n \in \mathcal{N}(m)$ and $m \in (1, M)$:
$$R_{mn} := \left[\prod_{k \in \mathcal{N}(m)\setminus n} \operatorname{sgn}(Q_{km})\right] \min_{k \in \mathcal{N}(m)\setminus n} |Q_{km}| \tag{2}$$

3. *Bit nodes*: Adding minimum values and LLR input values. For every $m \in \mathcal{M}(n)$ and $n \in (1, N)$:
$$Q_{nm} := L(x_n|y_n) + \sum_{k \in \mathcal{M}(n)\setminus m} R_{kn} \tag{3}$$

4. *Pseudo-posteriori probabilities*: Determination of pseudo-posteriori probabilities. For every $n \in (1, N)$:
$$Q_n := L(x_n|y_n) + \sum_{k \in \mathcal{M}(n)} R_{kn} \tag{4}$$

5. *Hard decisions*: Making trial, hard decisions. For every $n \in (1, N)$:
$$\hat{x}_n := \begin{cases} 1 & \text{gdy } Q_n < 0 \\ 0 & \text{gdy } Q_n \geq 0 \end{cases} \tag{5}$$

6. *Verification of control equations*:
$$\mathbf{H}\hat{x}^T = 0 \tag{6}$$

7. *Another iteration*: If the control equations have been met, decoding is terminated with $\hat{x}_n$ as an outcome. Otherwise, the next iteration of decoding begins, starting from the (2) control nodes, unless the iteration limit is exceeded.

where:

$:=$ meaning is "becomes",
$\mathbf{x} = [x_1, x_2, \ldots, x_n]$—code vector,
$\mathbf{y} = [y_1, y_2, \ldots, y_n]$—received vector,
Q_{nm}—credibility LLR value from the n-th bit vertex to the m-th Tanner graph control vertex,
$L(x_n|y_n)$—LLR a priori probabilities for the n-th bit,
$(1, N)$ is a set of integers between 1 and N,
$\mathcal{N}(m)$—a set of column indexes in the parity check matrix $\mathbf{H}$ containing one in the m-th row,
$\mathcal{M}(n)$—a set of row indexes in the parity check matrix $\mathbf{H}$ containing one in the n-th column,
R_{mn}—message from the m-th control vertex to the n-th bit vertex of Tanner graph,
$\hat{x}_n$—decoded vector.

It is known [21] that significantly improved decoding performance can be obtained by adding a normalizing parameter to Equation (2). Equation (2) will then take the form (7), where $\beta > 1$ is a normalizing parameter, or equivalently form (8), where $\alpha = 1/\beta$ and $0 < \alpha < 1$.

$$R_{mn} := \left[\prod_{k \in \mathcal{N}(m)\setminus n} \operatorname{sgn}(Q_{km})\right] \frac{\min_{k \in \mathcal{N}(m)\setminus n} |Q_{km}|}{\beta} \tag{7}$$

$$R_{mn} := \left[\prod_{k \in \mathcal{N}(m)\setminus n} \operatorname{sgn}(Q_{km})\right] \min_{k \in \mathcal{N}(m)\setminus n} |Q_{km}| \times \alpha \tag{8}$$

3. New Concepts of Technology Mapping of QC-LDPC Decoder Oriented to FPGA

3.1. Construction of the QC-LDPC Decoder with a Distributed Control System

The block diagram of the QC-LDPC irregular decoder is shown in Figure 3. Initially, the decoder receives a priori LLR values $L(x_n|y_n)$, which are stored in memory. The data is propagated between modules in a portion of P messages parallel in a data bus. The received data is forwarded to the initialization module and then P messages in the bus are re-positioned by cyclically shifting to the right using the Shift Right (SR) module. The number of positions to shift the data vector is defined in the offset memory S (Shift), which contains the stored shift values, corresponding to **P** submatrices of **H**. The shifted message vectors are stored in Q_{nm} memory, according to the Equation (1).

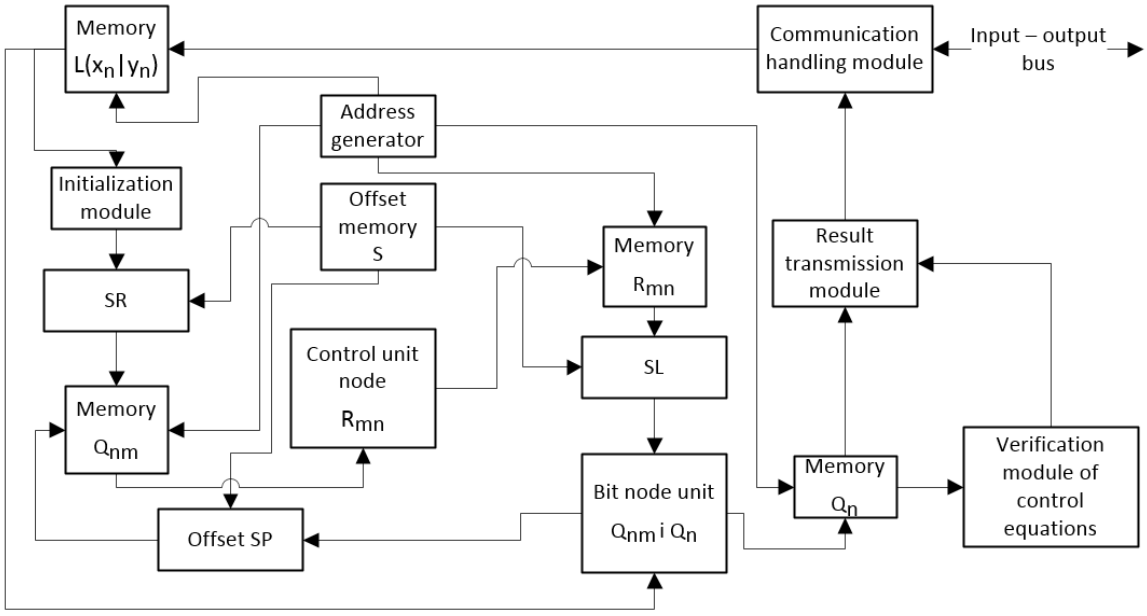

Figure 3. Block diagram of the QC-LDPC irregular code decoder.

The unit computing control node messages reads data from the Q_{nm} memory and determines the minimum value and the sign according to the Equation (2). The obtained result can be modified by the normalizing parameter α according to the Equation (8). The computed control node messages are saved to the R_{mn} memory.

The unit calculating bit node messages reads data from the R_{mn} memory and then cyclically shifts them to the left using the SL module. The unit computes the sums of appropriately chosen messages stored in R_{mn} and $L(x_n|y_n)$ memories, in accordance with the (3) equation. Formulas (3) and (4) are similar, therefore the Q_{nm} and Q_n are computed simultaneously in the bit node unit. The obtained results are stored in memories Q_{nm} and Q_n respectively.

In the next stage, the control equations are verified according to (6). Control equations are checked in verification module of control equation. If all control equations have been met, the result transmission module begins reading data from the Q_n memory by transferring it in the correct order to the communication handling module. Otherwise, all calculations are repeated, starting with the control unit node. Decoding can be interrupted if the assumed maximum number of iterations has been reached. Each of the presented elements of the QC-LDPC decoder has its own control unit that is responsible for its proper operation.

A typical method for implementing a control unit of an LDPC decoder is to use a global controller, which controls the operation of the entire decoder. The disadvantage of this solution is the high level of complexity of the controller and the possible occurrence of clock skew phenomena.

The proposed decoder uses a distributed control system. Elements: "communication handling module", "initialization module", "control unit node", "bit node unit", "verification module of control equations" and "result transmission module" in Figure 3 have their own built-in control units. The control of each element is activated by the preceding control unit as described by the decoder. The first control unit in the communication handling module is activated by an external signal informing it about the start of the transfer of a new, received data vector. A similar principle is used in systems with so-called Token Passing, e.g., the IEEE 802.5 Token Ring [22] standard developed in the 70s by IBM. This allowed for a significant simplification of the construction of individual control units and their better adaptation to the needs of a given element. The distributed control system is also less susceptible to clock skew phenomena.

Figure 4 shows a simplified layout of the QC-LDPC decoder, which indicates in what order the Token is activated. The red arrows indicate the stages of running the distributed control system. The dotted arrows are optional and, depending on the decision made by the verification module of control equations, the Token will be passed to the result transmission module or the unit for calculating control nodes.

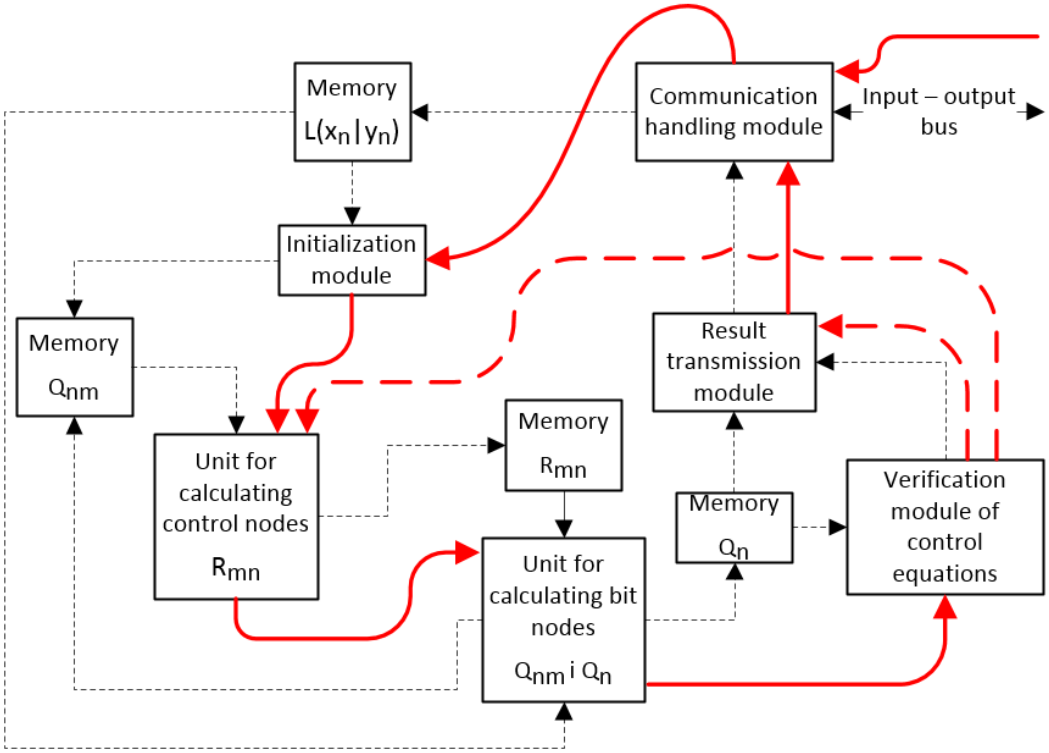

Figure 4. Handing over the Token in a distributed control system.

Memory controlling and addressing is performed by control units located in the other elements of the decoder. Since more than one element uses every message memory, an attention should be paid to the address propagation. For this purpose, a multiplexer is used as shown in Figure 5. When implementing the multiplexer in a hardware description language (e.g., Verilog), it must be ensured that the selection of the address takes place depending on which Trigger was last to change its state (edge trigger).

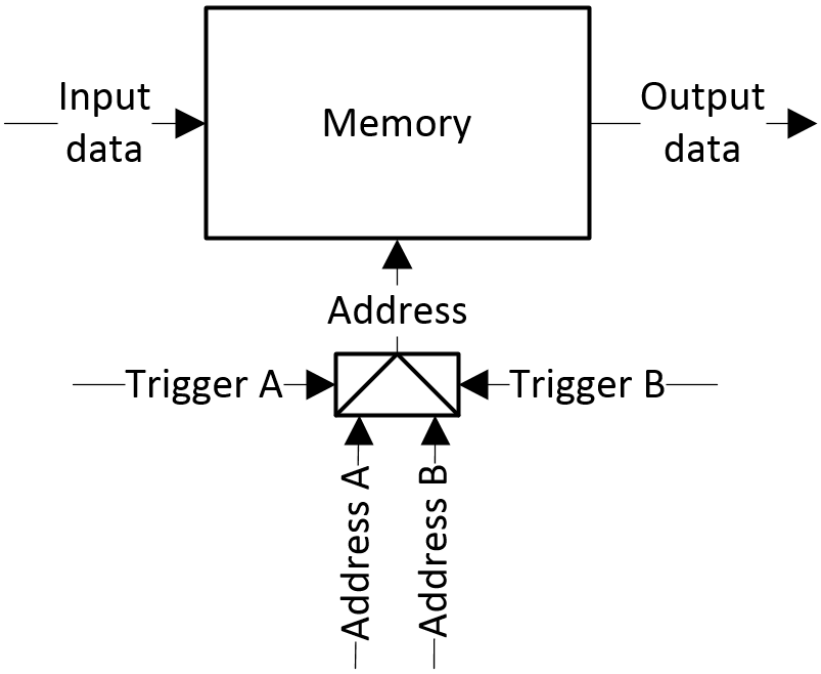

Figure 5. Memory addressing by several control units.

As mentioned before, in the developed distributed control system, each system element is activated by the preceding element, by detecting a change in the "Trigger X" signal, as shown in Figure 6. After finishing the current iteration tasks, the control unit activates the next control unit with the "Trigger Y" signal, at the same time deactivating itself. Every unit of the control system works only when the "Enable" signal is active.

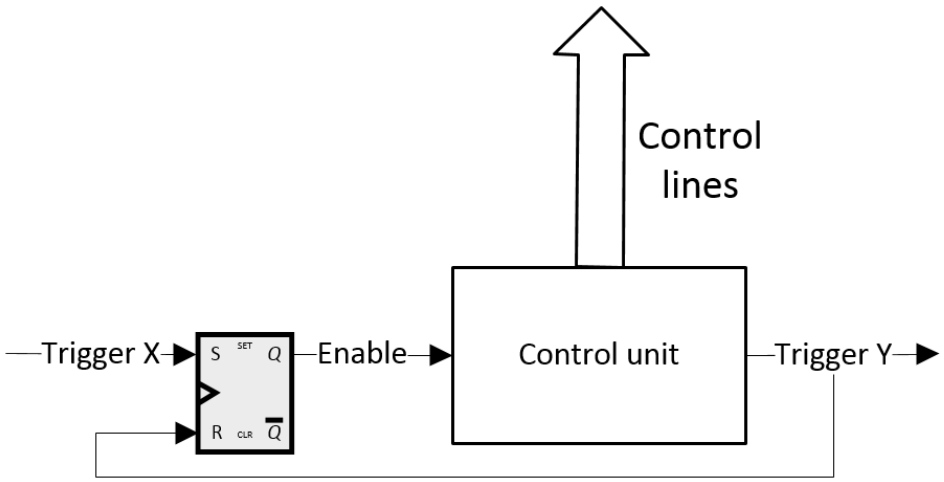

Figure 6. Enabling and disabling the control system in the decoder element.

3.2. Implementation of the Normalization Module

The block diagram of the unit calculating the control nodes messages R_{mn}, with normalization elements of the NMS algorithm, is presented in Figure 7. The data read from memory is converted from the two's complement format (convenient for additions in bit node units) to the Sign and Module (SM) format. Every data word contains P messages, represented by B-bit fixed point numbers, that are separated and delivered to appropriate Min-Sign modules. Every Min-Sign module has d_c inputs, for messages corresponding to at most d_c check node edges. The value of d_c is also equal to the largest number of non-zero elements in any row of **H**. The DMUX and MUX modules deliver the messages

from memory to inputs of appropriate Min-Sign modules. Counters addressing the DMUX and MUX are part of the control system unit. When the packet of data is smaller in a given computation cycle than the maximum d_c, the other outputs of the DMUX are set to the maximum (positive) value, which is transparent for the minimum operation as well as the sign-product operation.

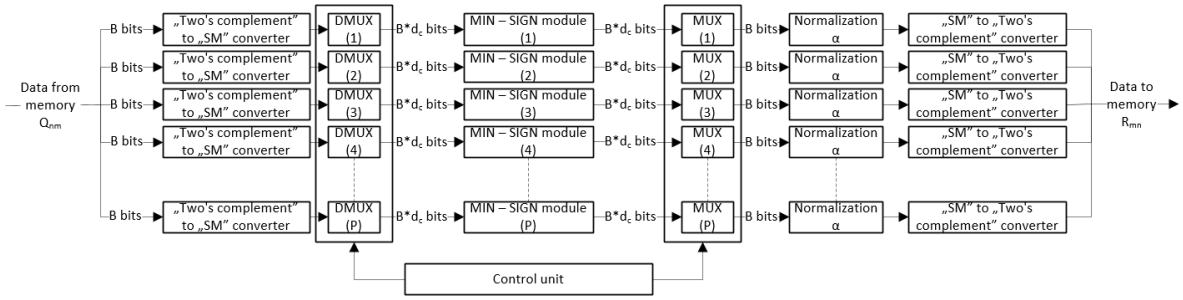

Figure 7. Block diagram of the unit calculating the control nodes messages R_{mn}.

The values of the minimum and the sign are determined in P separate Min-Sign modules operating in parallel, then normalized by the α parameter, according to (8). After multiplexing in MUX, results are converted back from the SM representation to the two's complement format and saved in the R_{mn} memory.

The typical hardware-efficient implementation of the multiplication of a fixed-point number by a constant coefficient (α) can be presented in the form of a module consisting of shifters and adders. In this method, $\alpha < 1$ is expressed as:

$$\alpha = \alpha_1 2^{-1} + \alpha_2 2^{-2} + \ldots + \alpha_B 2^{-B} \tag{9}$$

where $\alpha_1, \alpha_2, \ldots \in \{0, 1\}$. Since multiplication by 2^{-b} is equivalent to shifting the fixed-point binary representation by b bits towards LSB, and $\alpha_1, \alpha_2, \ldots$ are constants, multiplication of a message m by α can be realized by summing these b-shifts of message m, for which $\alpha_b = 1$ in (9), $b = 1, 2, \ldots, B$.

For example, multiplication by $\alpha = 0.75 = 0.5 + 0.25$ can be implemented making use of a single adder, while multiplication by 0.5 and 0.25 is realized by shifting the fixed-point number by one and two bits, respectively. Performing this multiplication using shift registers and B-bit adders results in a truncation error, because of shifting-out the least significant bits. For data buses smaller than 6-bit, the truncation error can be quite significant, making other implementation more feasible. Therefore we propose another approach, which is oriented to the LUT-based FPGA.

The typical FPGAs chip contains an array of programmable logic blocks (ALM—Adaptive Logic Module in the case of Intel devices) and a hierarchy of reconfigurable interconnections that allow the blocks to be wired according to the specific project needs [23]. The ALM block consists of combinational logic (LUT), adders and registers [24]. The design of the ALM unit is shown in Figure 8. Each LUT has a maximum of 8 inputs and can be configured to realize any logic (binary) function.

Therefore, when implementing the normalization module in an FPGA structure, it is possible to perform a low-precision normalization in the form of a direct mapping of the normalizing function into FPGA logic resources, that is LUTs. Figure 9 presents Karnaugh maps for several parameters of α, with 4-bit precision. The maps are dependent only on the module of the number—sign bits are processed independently. It should be noted that multiplication by 0.625, 0.6875, 0.75 and 0.8125 values can be implemented using combinational logic (LUTs) and registers. A single LUTs is enough for a 4-bit precision. The $\alpha = 1$ parameter is also considered in Figure 9, which corresponds to QC-LDPC decoding without normalization. During the research, numerous variants of the Karnaugh maps were verified. The article presents one of the best Karnaugh map for the tested control matrix **H**.

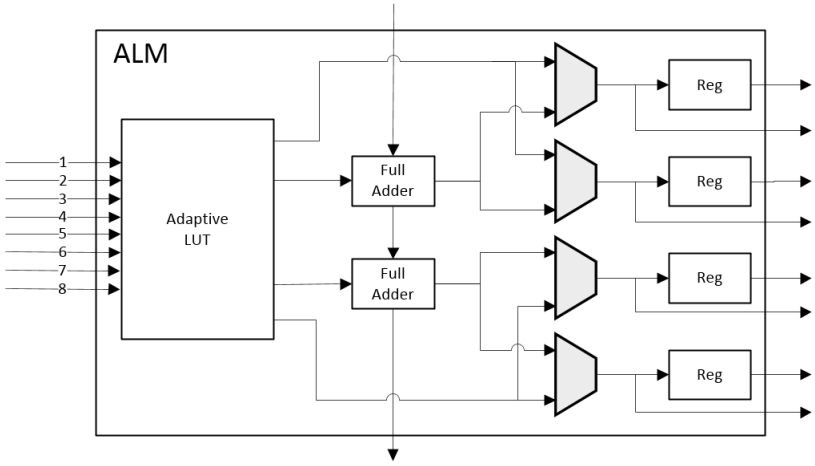

Figure 8. Design of the ALM unit in the Cyclone V system [24].

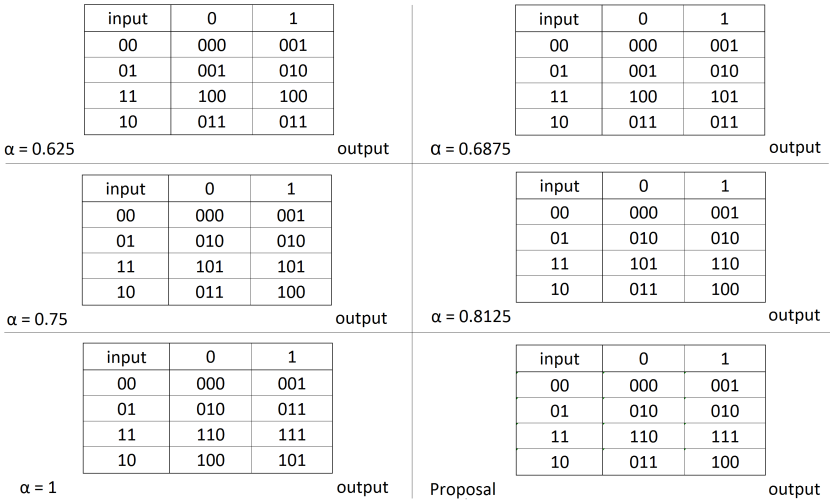

Figure 9. Karnaugh maps for several α parameters and 4-bit bus.

However, since the linear normalization is just an approximated method of belief calculation, we assumed that it is possible that application of another (non-linear) normalization functions, can result in not worse, possibly improved decoding correction performance. An example is the function presented in the last set of Karnaugh maps, labeled "Proposal", which is an arbitrarily chosen map, experimentally shown to give the best results, at least for the case codes that we have experimented with. This map is similar to α-normalization, but not exactly the same, and—as will be shown—results in an improved decoding performance of the implemented decoder. Moreover, this map can still be implemented in a single LUT of the FPGA. A logic function resulting from the synthesis of Karnaugh maps can be implemented directly in an LUTs. Implementation of the normalization expressed as a logic function is beneficial due to the direct fit into the FPGA architecture with LUTs.

Figure 10 presents graphically the dependence between 3-bit input module (represented by an integer in the range $0 \ldots 7$) and 3-bit output of normalization modules, for a few investigated normalizations, including the "Proposal" function that showed the best correction performance in our simulation experiments, as will be presented in the next section.

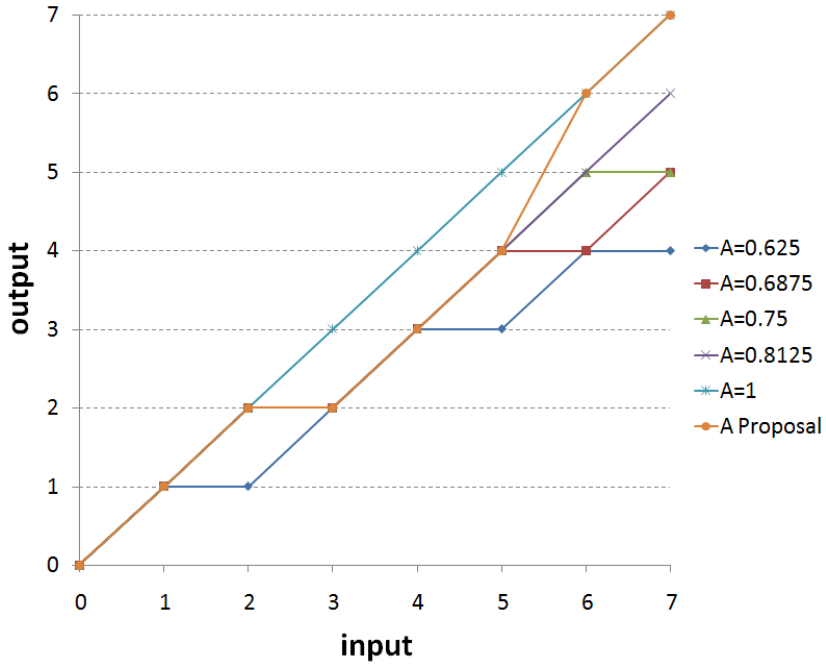

Figure 10. Different normalization functions investigated for a 4-bit precision (3-bit module) case.

4. Experimental Results

A special test environment was developed for the purpose of experimental research, the block diagram of which is presented in Figure 11. The environment consists of three key elements: computer (PC), microcontroller (eval-board STM32F4DICOVERY) and FPGA system (eval-board with Cyclone V device). The PC computer runs a system simulation software developed in Python, modeling random data generator, LDPC encoder, Binary Phase Shift Keying (BPSK) modulator/demodulator, AWGN (Additive White Gaussian Noise) channel and LDPC decoder. The encoded and modulated data is disturbed using the AWGN channel model, with noise level according to the variable Signal-to-Noise Ratio (SNR). The erroneous data is then corrected (decoded) by LDPC decoding software. Such a testing system allows verification of the correction properties of the LDPC code with given parity check matrix **H**. The results of the simulation the form of Bit Error Ratio (BER) vs SNR in an AWGN channel can be used as a reference chart for hardware implementations of the LDPC decoder.

The developed computer software is also responsible for generating description of QC-LDPC decoders in a Hardware Description Language (Verilog) for the FPGA chip. The generator as input parameters needs the parity check matrix **H**, specified by matrix size *MtimesN*, submatrix size *P*, location of nonzero submatrices and corresponding cyclical shift values. The generator creates all files (in Verilog) that are necessary to build a project in a Quartus environment. Additional parameters for the generator are:

- FPGA type and family (e.g., Cyclone V, 5CSEBA6U23I7);
- bit-resolution of decoder input—a priori LLRs (e.g., 4-bits);
- maximum number of decoder iterations (e.g., 50);
- type of algorithm used (e.g., Min-Sum or Normalized Min-Sum with chosen normalization).

A set of test data consisting of coded vectors (without interference) and distorted data vectors can be generated by the developed software, which allows verification of the QC-LDPC decoder FPGA implementation. A graphical interface of the whole environment was created with C#, while the microcontroller software was written in the C language. The aim of the microcontroller is to distribute the data, the clock and other control signals to the FPGA.

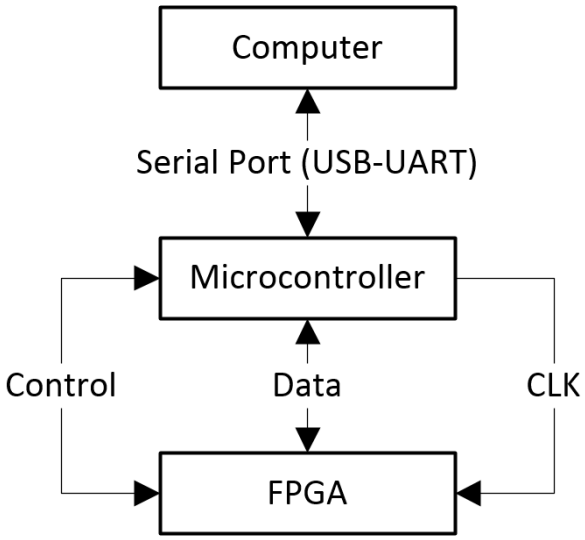

Figure 11. Test environment developed.

Making use of the environment, we can easily compare different constructions and configurations of the hardware decoder, reporting the hardware utilization as well as experimentally obtained error correction performance curves.

Table 1 shows the hardware requirements, obtained as a result of logic synthesis of a few selected configurations of the QC-LDPC decoder. It can be observed that as the data bus size increases, the hardware utilization for ALM units and memory bits also increases. Meanwhile, the normalization choice, including the proposed non-linear variant ("Proposal"), has no impact on the decoder hardware utilization. Therefore, the Proposal can be applied without any implementation overhead.

Table 1. Hardware utilization for selected versions of the QC-LDPC decoder.

	Hardware Utilization	
Bus size, algorithm	ALM	Memory bits
3-bits, Min-Sum	8071	28,312
4-bits, Min-Sum	13,531	35,864
5-bits, Min-Sum	16,464	43,416
4-bits, Normalized Min-Sum $\alpha = 0.625$	13,557	35,864
4-bits, Normalized Min-Sum $\alpha = 0.6875$	13,561	35,864
4-bits, Normalized Min-Sum $\alpha = 0.75$	13,572	35,864
4-bits, Normalized Min-Sum $\alpha = 0.8125$	13,564	35,864
4-bits, Normalized Min-Sum Proposal	13,572	35,864

Figure 12 shows the observed error correction performance of the QC-LDPC decoder implemented in the Cyclone V (FPGA), illustrating dependence on the message (belief) precision. The Bit Error Rate (BER) and Frame Error Rate (FER) curves for the 3-bit data reflect very poor correction performance for the presented SNR range. Meanwhile, the 4-bit and 5-bit precision decoders correct errors with effectiveness increasing with SNR. Differences in BER and FER curves between 4- and 5-bit cases depend somewhat on the SNR, but they are not very significant. Therefore, in general a 4-bit bus is recommended, because of its better correction properties than the 3-bit solution and less hardware utilization in relation to the 5-bit solution. All tests were performed for the parity check matrix **H** shown in Figure 2 and the maximum number of iterations of 15. Figure 12 as a reference provides the corresponding BER and FER curves obtained from computer a simulation with the floating-point precision BP decoding.

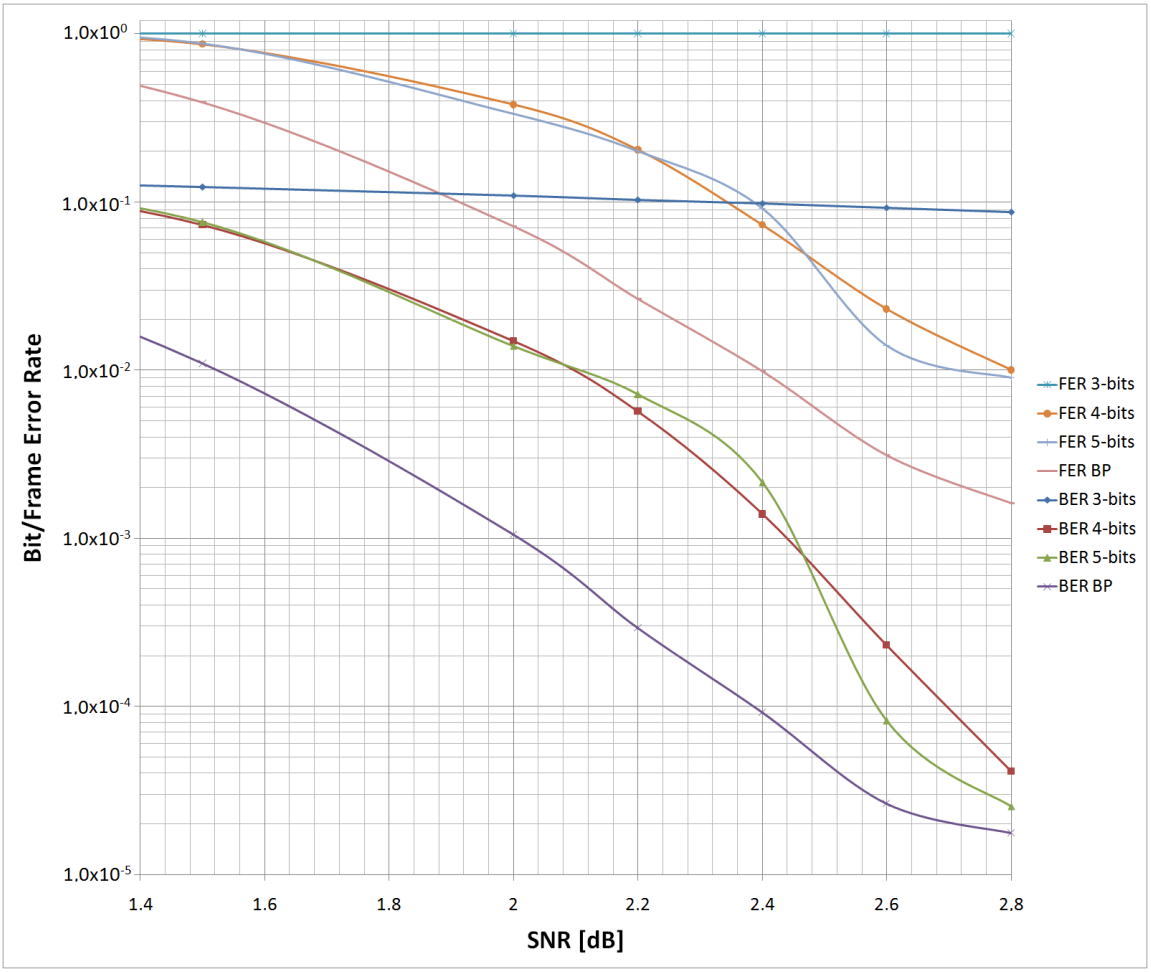

Figure 12. BER and FER chart for QC-LDPC decoder depending on data bus size.

The next simulation results provide an experimental insight into the normalization method of the NMS algorithm. Figure 13 presents the BER and FER results in the examined SNR range for the Min-Sum algorithm ($\alpha = 1$) and a range of variants of the NMS algorithm, with the recommended 4-bit precision. It can be observed that in general, the Min-Sum algorithm has performance inferior to the NMS algorithm with optimized α, with $\alpha = 0.8125$ being the optimal value in this case. Meanwhile, Min-Sum outperforms NMS with some other values of α.

However, it can be also observed that the use of the "Proposed" solution gives correction performance even better than NMS with the best $\alpha = 0.8125$. The proposed nonlinear normalization method makes it possible to achieve to achieve significantly improved correction performance than the optimized NMS. Meanwhile, it can be well fitted into FPGA resources, with nonlinear normalization still implemented in a single LUTs. Therefore, we achieved a modified NMS algorithm implementation, well suited for implementation in FPGAs without any hardware overhead, but with an improved correction performance.

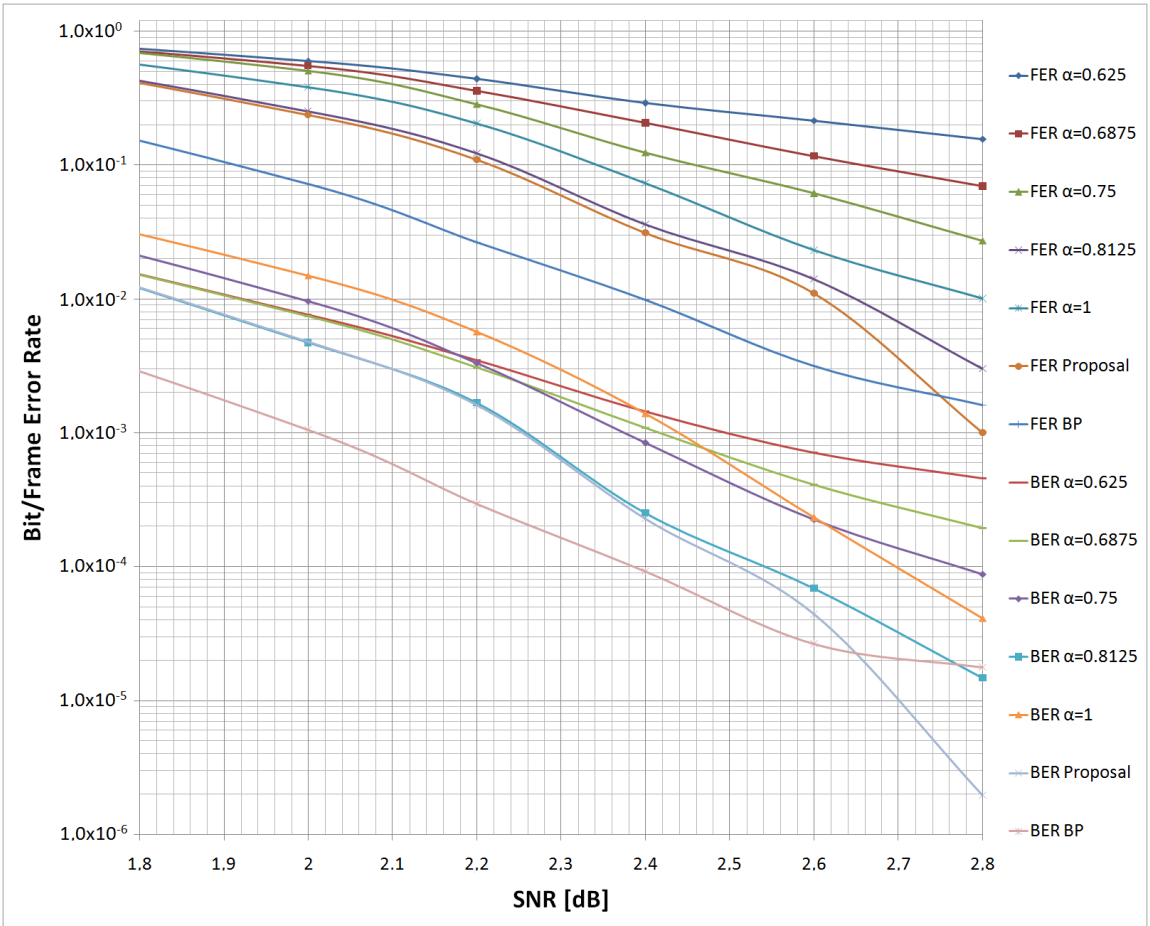

Figure 13. BER and FER chart of the QC-LDPC decoder for the Min-Sum algorithm and variants of the Normalized Min-Sum algorithm.

5. Conclusions

The article presents a hardware (FPGA) implementation of the irregular QC-LDPC decoder. A few variants of decoding algorithm and precisions have been investigated. It is shown that it is beneficial to use a 4-bit precision, characterized by a better correction performance in comparison with the 3-bit precision, and hardware resources savings in comparison with 5-bit precision. The use of 4-bit buses does not lead to a significant deterioration in the correction performance in comparison with the decoder with 5-bit buses.

The novel idea presented in this article is a method of implementing the normalization of the NMS algorithm, enabling effective technological mapping into FPGA structures. The proposed method of nonlinear normalization with a selected logic function makes it possible to obtain a better correction performance in comparison to standard, well-known NMS solutions. A comparison of the hardware resources used for the different decoder variants indicates a slight increase in the number of ALM units used for the NMS algorithm and the proposed algorithm in comparison with the Min-Sum algorithm.

Particularly noteworthy is the interdisciplinary nature of the work presented. The use of the capabilities of a personal computer, a microcontroller and an FPGA system perfectly fits the idea of Cyber-Physical Systems, which combine issues in the fields of electronics, computer science and telecommunications. The developed test environment not only allows for automating many activities (e.g., generating new decoders, carry out tests) but also significantly expands the research capabilities. The use of a distributed control system allows for significant simplification of the construction of the QC-LDPC decoder as well as the control system itself.

Future work will focus on optimizing the developed systems in terms of energy consumption. The idea of a distributed control system provides many new possibilities, including an introduction of Clock Gating for different parts of the decoder, which will also be a further direction of work. It seems that it is possible to develop a decoder which, while ensuring the assumed correction parameters, can become more energy-efficient. The essence of these ideas is to reduce the dynamic power consumption by using Clock Gating methods in a way that best suits QC-LDPC decoders.

Author Contributions: Conceptualization, M.K., W.S. and D.K.; methodology, M.K., W.S. and D.K.; software, M.K., W.S. and D.K.; validation, M.K., W.S. and D.K.; formal analysis, M.K., W.S. and D.K.; investigation, M.K., W.S. and D.K.; resources, M.K., W.S. and D.K.; data curation, M.K., W.S. and D.K.; writing—original draft preparation, M.K., W.S. and D.K.; writing—review and editing, M.K., W.S. and D.K.; visualization, M.K., W.S. and D.K.; supervision, M.K., W.S. and D.K.; project administration, M.K., W.S. and D.K.; funding acquisition, M.K., W.S. and D.K. All authors have read and agreed to the published version of the manuscript.

Funding: The study was partially supported by the Polish Ministry of Science and Higher Education.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gallager, R.G. *Low-Density Parity-Check Codes*; MIT Press: Cambridge, MA, USA, 1963.
2. MacKay, D.J.C. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inf. Theory* **1999**, *45*, 399–431. [CrossRef]
3. Nurellari, E. LDPC Coded OFDM and It's Application to DVB-T2, DVB-S2 and IEEE 802.16e. Master's Thesis, Eastern Mediterranean University, Famagusta, North Cyprus, 2012.
4. Darabiha, A.; Carusone, A.C.; Kschischang, F.R. Power Reduction Techniques for LDPC Decoders. *IEEE J. Solid-State Circuits* **2008**, *43*, 1835–1845. [CrossRef]
5. Galli, S. On the Fair Comparison of FEC Schemes. In Proceedings of the 2010 IEEE International Conference on Communications, Cape Town, South Africa, 23–27 May 2010; pp. 1–6.
6. Pisek, E.; Rajan, D.; Cleveland, J.R. Trellis-Based QC-LDPC Convolutional Codes Enabling Low Power Decoders. *IEEE Trans. Commun.* **2015**, *63*, 1939–1951. [CrossRef]
7. Chen, Z.; Zhao, X.; Peng, X.; Zhou, D.; Goto, S. A high-parallelism reconfigurable permutation network for IEEE 802.11n??802.16e LDPC decoder. In Proceedings of the 2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Kanazawa, Japan, 7–9 January 2009; pp. 85–88.
8. Mukherjee, M.; Kumar, P.; Matam, R. A PHY-layer framework of multirate transmission for Ultra-Dense Networks in 5G. In Proceedings of the 2015 International Conference on Communications, Signal Processing, and Their Applications (ICCSPA'15), Sharjah, UAE, 17–19 February 2015; pp. 1–6.
9. Tanner, R. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory* **1981**, *27*, 533–547. [CrossRef]
10. Li, J.; Liu, K.; Lin, S.; Abdel-Ghaffar, K. Algebraic Quasi-Cyclic LDPC Codes: Construction, Low Error-Floor, Large Girth and a Reduced-Complexity Decoding Scheme. *IEEE Trans. Commun.* **2014**, *62*, 2626–2637. [CrossRef]
11. Wang, Z.; Cui, Z.; Sha, J. VLSI Design for Low-Density Parity-Check Code Decoding. *IEEE Circuits Syst. Mag.* **2011**, *11*, 52–69. [CrossRef]
12. Tasdighi, A.; Banihashemi, A.H.; Sadeghi, M.R. Efficient Search of Girth-Optimal QC-LDPC Codes. *IEEE Trans. Inf. Theory* **2016**, *62*, 1552–1564. [CrossRef]
13. Liu, X.; Xiong, F.; Wang, Z.; Liang, S. Design of Binary LDPC Codes with Parallel Vector Message Passing. *IEEE Trans. Commun.* **2018**, *6*, 1363–1375. [CrossRef]
14. Chen, J.; Dholakia, A.; Eleftheriou, E.; Fossorier, M.P.C.; Hu, X.Y. Reduced-Complexity Decoding of LDPC Codes. *IEEE Trans. Commun.* **2005**, *53*, 1288–1299. [CrossRef]
15. Zhao, J.; Zarkeshvari, F.; Banihashemi, A.H. On Implementation of Min-Sum Algorithm and Its Modifications for Decoding Low-Density Parity-Check (LDPC) Codes. In Proceedings of the 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Valencia, Spain, 6–8 June 2018; pp. 1–5.

16. Hailes, P.; Xu, L.; Maunder, R.G.; Al-Hashimi, B.M.; Hanzo, L. A survey of FPGA-based LDPC decoders. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1098–1122. [CrossRef]
17. Andrade, J.; George, N.; Karras, K.; Novo, D.; Pratas, F.; Sousa, L.; Ienne, P.; Falcao, G.; Silva, V. Design Space Exploration of LDPC Decoders Using High-Level Synthesis. *IEEE Access* **2017**, *5*, 14600–14615. [CrossRef]
18. Liao, Y.C.; Lin, C.; Chang, H.C.; Lin, S. A (21150, 19050) GC-LDPC Decoder for NAND Flash Applications. *IEEE Trans. Circuits Syst. Regul. Pap.* **2019**, *66*, 1219–1230. [CrossRef]
19. Zhang, K.; Huang, X.; Wang, Z. High-Throughput Layered Decoder Implementation for Quasi-Cyclic LDPC Codes. *IEEE J. Sel. Areas Commun.* **2009**, *27*, 985–994. [CrossRef]
20. Sułek, W. Protograph based low-density parity-check codes design with mixed integer linear programming. *IEEE Access* **2019**, *7*, 1424–1438. [CrossRef]
21. Chen, C.; Xu, Y.; Ju, H.; He, D.; Zhang, W.; Zhang, Y. Variable Correction for Min-Sum LDPC Decoding Applied in ATSC3.0. *IEEE Trans. Commun.* **2005**, *53*, 549–554.
22. IEEE. *IEEE Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Part 5: Token Ring Access Method and Physical Layer Specifications*; ANSI/IEEE Std 802.5-1998 (ISO/IEC 8802-5); IEEE: Piscataway, NJ, USA, 1998; pp. 1–260.
23. Kubica, M.; Kania, D. Technology mapping oriented to Adaptive Logic Modules. *Bull. Pol. Acad. Sci. Tech. Sci.* **2019**, *67*, 947–956.
24. Intel. *Cyclone V Device Overview, CV-51001*; Intel: Santa Clara, CA, USA, 2018.