electronics

Article

A Flexible FPGA-Based Stochastic Decoder for 5G LDPC Codes

Sivarama Prasad Tera *, Rajesh Alantattil and Roy Paily

Department of Electronics and Electrical Engineering, Indian Institute of Technology, Guwahati 781039, India; rajesha@iitg.ac.in (R.A.); roypaily@iitg.ac.in (R.P.)
* Correspondence: sivarama@iitg.ac.in

Abstract: Iterative stochastic decoding is an alternative to standard fixed-point decoding of low-density parity check (LDPC) codes that can be used to minimize inter-node routing. A flexible field-programmable gate array (FPGA)-based stochastic decoding (SD) hardware architecture is presented in this paper. The architecture is designed to decode different code rates of LDPC codes that comply with the fifth-generation (5G) New Radio (NR) standard. This decoder's runtime flexibility is desirable as it switches to a better-performing code rate automatically based on the channel conditions without the extra time needed to reprogram the FPGA. An offline design method is implemented to generate the hardware description language (HDL) code description of the decoder for the required code rate set, which is further synthesized and integrated into a Xilinx Kintex-7-series FPGA board to determine the hardware resource utilisation and processing throughput. Synopsys design tools were employed during both the simulation and synthesis stages in combination with TSMC 65 nm CMOS standard cell technology to facilitate comparative analysis. Compared with state-of-the-art designs, the proposed architecture reduces hardware utilization by up to 26% and increases energy efficiency by 52%.

Keywords: 5G NR standard; error-correcting codes; low-density parity check codes (LDPCs); stochastic decoding (SD); field-programmable gate array (FPGA)

check for updates

Citation: Tera, S.P.; Alantattil, R.; Paily, R. A Flexible FPGA-Based Stochastic Decoder for 5G LDPC Codes. *Electronics* **2023**, *12*, 4986. https://doi.org/10.3390/electronics12244986

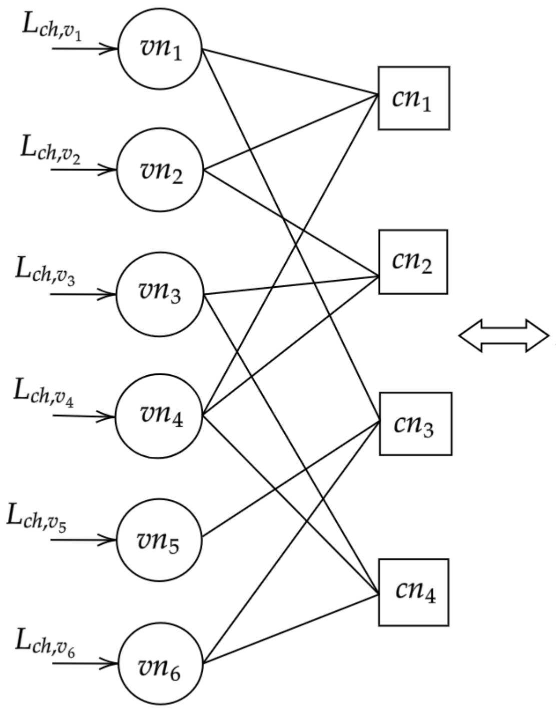
Academic Editor: Alexander Barkalov

Received: 19 October 2023
Revised: 4 December 2023
Accepted: 7 December 2023
Published: 13 December 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Low-density parity check codes (LDPCs) [1] have become one of the essential channel codes in many communication standards, such as DVB-S2 [2], IEEE 802.11 (WiFi) [3], IEEE 802.16e (WiMax) [4], and including fifth-generation (5G) wireless technology [5], because of their higher error correction capabilities, which are close to Shannon's limit [6,7]. In 5G New Radio (NR) specifications, quasi-cyclic (QC) LDPC codes are selected as the channel coding scheme for data channels to achieve high throughput and low latency [8]. These QC LDPC codes have adopted two base graph matrices (BGMs): H_{b1} and H_{b2}, and fifty-one lifting sizes or expansion factors z_c to support various code rates [8]. H_{b1} has the dimension of 46 block rows and 68 block columns, and it supports code rates ranging from 1/3 to 8/9. H_{b2} has a dimension of 42 block rows and 52 block columns and supports code rates from 1/5 to 2/3. The PCM H is represented and constructed from its BGM H_b [9]. A binary (N, K) LDPC code in 5G is characterized by the null space of parity check matrix (PCM) H with dimension $M \times N$ over GF(2). The PCM is also visualized graphically as a bipartite Tanner graph [10,11]. A set of M parity or check nodes of the Tanner graph represent the rows of H, and a set of N bit or variable nodes of the Tanner graph represent the columns of H. The decoding of LDPC codes is done iteratively using the message-passing algorithm on its bipartite Tanner graph [10]. For an example (6, 2) LDPC code, parity check matrix (PCM) H and its Tanner graph are shown in Figure 1. A Tanner graph comprises two types of nodes, one is bit or variable nodes (VNs) VN_1, VN_2, VN_3, VN_4, VN_5, VN_6 and the other is parity or check nodes (CNs) CN_1, CN_2, CN_3, CN_4. These nodes are connected by bidirectional edges based on the number of ones in the PCM of the code.



$$H = \begin{array}{cccccc|c} vn_1 & vn_2 & vn_3 & vn_4 & vn_5 & vn_6 & \\ 1 & 1 & 0 & 1 & 0 & 0 & cn_1 \\ 0 & 1 & 1 & 1 & 0 & 0 & cn_2 \\ 1 & 0 & 0 & 0 & 1 & 1 & cn_3 \\ 0 & 0 & 1 & 1 & 0 & 1 & cn_4 \end{array}$$

Figure 1. Tanner graph of $(6,2)$ code.

Runtime flexibility of the decoder is desirable for decoding received messages associated with different BGMs of multiple code rates at runtime [12]. Hence, the decoder dynamically switches between a given set of LDPC code rates with diverse BGMs. This decoder has various commercial applications, such as switching to a better-performing code rate automatically based on the channel conditions [13] without the extra time needed to reprogram the FPGA. Another application is to eliminate the re-synthesis of the FPGA to test the performance of different code rates.

The specifications of 5G NR show that the BGM H_{b1} can support a vast range of code rates from 1/3 to 8/9. In BGM, these code rates generate irregular node degrees. As a result of the irregular node degrees of BGM, complex connections are formed between the nodes of the Tanner graph of BGM. In BGM, these code rates generate irregular node degrees. As a result of the irregular node degrees of BGM, complex connections are formed between the nodes of the Tanner graph of BGM. The complex connections result in a more complicated inter-node routing network for typical fixed-point LDPC decoders, which are based on the sum-product algorithm (SPA) [14] and the min-sum (MS) [15] algorithm. These decoders use multiple-bit fixed-point representations of channel logarithmic-likelihood ratio (LLR) message values as inputs to decoder nodes that need multiple paths to exchange these values between the nodes. This condition aggravates an already very difficult inter-node routing network problem. This is due to the fact that the architecture of the decoder nodes necessitates the use of a large number of FPGA resources.

One alternative solution is stochastic decoding (SD) [16]. In SD, initially, the channel LLRs are converted into equivalent channel probability values. After this conversion, these probability values are converted into an equivalent stochastic bit sequence by using a comparator, as shown in Figure 2. This conversion helps stochastic bit sequences, which need a single path between the nodes for exchange rather than the multiple paths needed for nodes in conventional fixed-point decoders. This significantly helps to reduce the inter-node routing complexity of the decoder [17]. The single bit-wise computations in stochastic variable node (SVN) and stochastic check node (SCN) units require simple logic units for the implementation. These stochastic LDPC decoders provide error correction capability comparable to traditional fixed-point decoders [18]. The error correction capability of an LDPC decoder is determined by the signal-to-noise power ratio per bit

E_b/N_o at the required bit error rate (BER), which is commonly stated in decibels. For the wireless personal area network (WPAN) [19] standard (672, 336) LDPC code, the suggested stochastic decoder in [18] has achieved a BER of 10^{-6} at an E_b/N_o value of 3.5 dB. This is in comparison to the normalized min–sum (NMS)-based decoder, which has achieved the same BER at an E_b/N_o value of 3.4 dB.

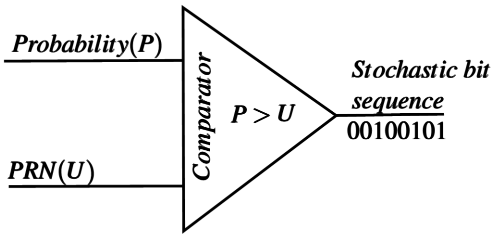


Figure 2. Schematic of comparator.

The number of node-interconnects (I) of a decoder is calculated from (1) to determine the routing complexity [20].

$$I = 2 \times N \times e_l \times w_v \tag{1}$$

N is codeword length, e_l is extrinsic message length, and w_v is column weight. For instance, 1/3-rate codeword length $N = 3808$ LDPC code having average column weight $w_v = 4.56$ is decoded using SPA and SD. The number of node-interconnects required for SPA and SD are 138,916 and 34,729, respectively. The extrinsic message length e_l = 1 for SD is responsible for the reduction of the number of node-interconnects in SD by a factor of four. On the other hand, the minimum value of e_l for fixed-point LDPC decoders that use SPA is determined to be 4.

Our contributions:

- This work's main contribution is to propose a new partially parallel decoder architecture for bit-wise stochastic decoding for 5G NR standard LDPC codes. This architecture was created for the LDPC code, which has a code word length of N = 3808 and code rates of $R = 1/3, 2/5, 1/2, 2/3, 3/4, 5/6$, and $8/9$ for BGM1.
- Our proposed automated design flow procedure enables runtime flexibility in the design. It creates an optimal FPGA-based stochastic LDPC decoder design for any selected code rate set. This approach helps to reduce the time needed to design hand-coded interconnections in hardware description language (HDL).

This paper is structured as follows. Section 2 introduces the basic concepts of stochastic decoding and its algorithm. Section 3 focuses on constructing BGM of 5G NR standard QC LDPC codes suitable for stochastic decoders. Section 4 provides architecture details. Design flow is discussed in Section 5. Section 6 discusses simulation results. Finally, the conclusion is noted in Section 7.

2. Preliminaries

2.1. Stochastic Bit-Sequence Generation

A stochastic approach to LDPC decoding was introduced in [21]. This decoding process converts the received channel probability values into a stochastic bit-sequence equivalent. In general, the LLR of the s-th symbol x_s of a code word is found from the s-th received channel symbol y_s at SVN_s, which is represented as L_{ch,v_s}. By considering the binary phase-shift keying (BPSK) modulation over an additive white Gaussian noise (AWGN) channel with a zero mean and variance σ^2, the channel LLR computed as $L_{ch,v_s} = 2 \times y_s/\sigma^2$. The channel probability is also known as the intrinsic message and is calculated as

$$P_{ch,v_s} = P(x_s = 1/y_s) = \left[\frac{exp(L_{ch,v_s})}{exp(L_{ch,v_s}) + 1} \right] \tag{2}$$

The term $exp()$ is an exponential function. The P_{ch,v_s} value is represented with W binary symbols and compared with W-bit pseudo-random number (PRN) $U \in [0,1]$ of the comparator [22], which varies with every clock cycle to generate an equivalent stochastic bit sequence of length $f = 8$, as shown in Figure 2. For example, $P_{ch,v_s} = 0.3125$ is represented as $W = 4$-bit fractional binary symbols $P = 0101$. At each clock cycle, a PRN generator based on a linear feedback shift register (LFSR) [23] produces a new $W = 4$-bit PRN U. Suppose $P > U$: the comparator output at that clock cycle is 1 and otherwise is 0. As shown in Table 1, after eight clock cycles, the comparator output is 00100101, which is the stochastic bit sequence belonging to $P_{ch,v_s} = 0.3125$. This sequence has a mean value of 3/8, which is close to $P_{ch,v_s} = 0.3125$ [24].

Table 1. Example of stochastic bit-sequence generation.

Clock Cycle (*k*)	*U*	u (*Real Value*)	Comparator Output (*P* > *U*)
0	1101	0.8125	0
1	0111	0.4375	0
2	0011	0.1875	1
3	0110	0.375	0
4	1001	0.5625	0
5	0010	0.125	1
6	1100	0.75	0
7	0100	0.25	1

2.2. Stochastic Decoding Algorithmic Description

A binary (N, K) LDPC code in 5G is characterized by the null space of parity check matrix (PCM) H with dimension $M \times N$ over GF(2). The PCM is also visualized graphically as a bipartite Tanner graph [10,11]. A set of M parity or check nodes of the Tanner graph represent the rows of H, and a set of N bit or variable nodes of the Tanner graph represent the columns of H. In stochastic decoding of LDPC code, the stochastic bit sequence of the channel probability values corresponds to their respective channel LLRs exchanged iteratively bitwise between the SCNs and the SVNs until the desirable codeword is found or the maximum iteration limit is reached [25]. To illustrate the simplification of the decoding process, a degree-3 SCN and a degree-3 SVN are adopted. Both SCN and SVN elements perform the bitwise modulo-2 arithmetic operations in SD. The steps are shown in the following:

(1) Initialization: Once the channel probability values are reached at SVNs, compare the W-bit fractional binary equivalent sequence of P_{ch,vn_j} with W-bit PRN U and initialize generated stochastic bit sequence to its corresponding SVNs vn_j.

(2) SCN update: As shown in Figure 3, SCN_2 is connected to three SVNs: SVN_2, SVN_3, and SVN_4. The arriving SCN-to-SVN single-bit extrinsic message for node SVN_4 is computed as

$$F_{c_2 \to v_4} = a.(1-b) + b.(1-a) \tag{3}$$

Here, a is the single-bit of stochastic bit sequence $\{a[k]\}, k = 0, \ldots, f-1$, belonging to its P_{ch,v_2}, which has been received as a message from SVN_2 to SCN_2 in the previous clock cycle, and b is the single-bit of stochastic sequence $\{b[k]\}$ belonging to its P_{ch,v_3}, which has been received as a message from SVN_3 to SCN_2 in a previous clock cycle.

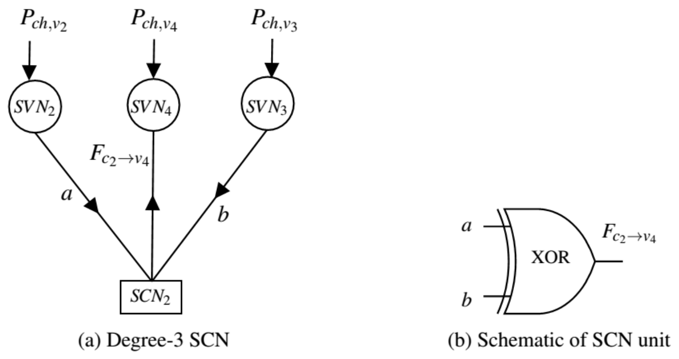


Figure 3. SCN and its implementation unit.

(3) SVN update: In Figure 4, SVN_4 is connected to three SCNs: SCN_1, SCN_2, and SCN_4. The arriving SVN-to-SCN single-bit extrinsic message for node SCN_2 is computed as

$$F_{v_4\to c_2} = \frac{e.d}{e.d + (1-e).(1-d)} \tag{4}$$

Here, e is the single-bit of stochastic bit sequence $\{e[k]\}$ belonging to the received message from SCN_1 to SVN_4 in the previous clock cycle, and d is the single-bit of stochastic bit sequence $\{d[k]\}$ belonging to the received message from SCN_4 to SVN_4 in a previous clock cycle.

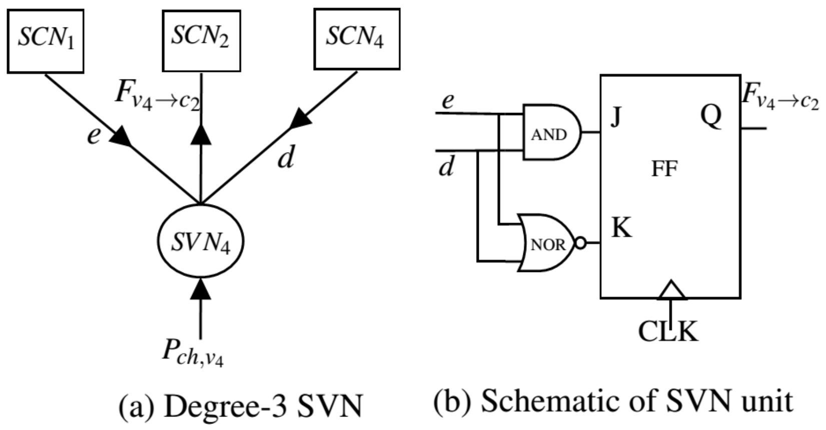


Figure 4. SVN and its implementation unit.

(4) Termination: This iterative process will stop if it reaches the maximum decoding cycle (DC) limit or all parity check equations are satisfied.

3. Construction of BGM in 5G NR Standard

The PCM H is represented and constructed from its BGM H_b [9]. The H_b has dimension $m_b \times n_b$, where $M = m_b \times z_c$ and $N = n_b \times z_c$. The entries of H_b are expanded with a $z_c \times z_c$ square sub-matrix in H, where z_c is known as the lifting size or expansion factor. The entry value '-1' of H_b is replaced by a zero matrix of dimension $z_c \times z_c$ in H, and the value '0' is replaced by the identity matrix of dimension $z_c \times z_c$, and another non-'-1' entry value $1 \le S_{i,j} \le z_c$, also known as the shift value, is replaced by circulant permutation matrix $I(S_{i,j})$. The subscripts i, j represent the row index and column index of the entry, respectively. The sub-matrix $I(S_{i,j})$ is obtained by each row of the identity matrix having a right-shift value of $S_{i,j}$ positions. The value of $S_{i,j}$ is calculated from the function (5). The value of $V_{i,j}$ is obtained from Tables 5.3.2-2 and 5.3.2-3 of 5G NR standard specification 3GPP TS 38.212 [8] according to the selected BGM and the set index i_{LS}.

In this paper, we constructed BGM H_{b1} of (N, K) LDPC code having a message or information block length $K = 1232$ and code rate $R = K/N = 1/3$. The five steps are listed below for constructing the H_{b1} with k_b as the length of the message block columns. The terms k_b of H_{b1} and K of H are related, as $K = k_b \times z_c$.

1. Select from the two BGMs: As per the specification of 3GPP TS 38.212 [8], since the code rate $R \geq 1/4$, BGM1 is selected.
2. Calculate the value k_b after selecting BGM: From the specification of 3GPP TS 38.212 [8], BGM1 has k_b = 22.
3. Find the expansion factor z_c : Select the minimum z_c from Table 5.3.2-1 [8] such that $k_b \times z_c \geq K$. For given $K = 1232$, k_b = 22, and z_c is calculated as $1232/22 = 56$.
4. Select the set index i_{LS}: After z_c is determined, the suitable shift coefficient matrix set from Table 5.3.2-1 [8] must be selected. Since $z_c = 56$, the set index i_{LS} = 3 is considered.
5. Compute the BGM entry values: Utilize the function (5) to determine the entry values $S_{i,j}$ by means of the modular z_c operation.

$$S_{i,j} = f(V_{i,j}, z_c) = \begin{cases} -1, & \text{if } V_{i,j} = -1; \\ mod(V_{i,j}, z_c), & \text{else} \end{cases} \tag{5}$$

6. Construct the PCM H: Substitute each entry of the BGM by the corresponding circulant permutation matrix or zero matrix of size $z_c \times z_c$ in H.

From Step 5, the first entry $h_{0,0}$ of H_{b1} is calculated using $mod(V_{0,0}, z_c)$ = mod (223, 56) = 55, where the selected $V_{0,0} = 223$ value is obtained from Table 5.3.2-2 [8] and corresponds to row index $i = 0$, column index $j = 0$, and set index $i_{LS} = 3$. Similarly, the remaining entry values are calculated by using a function (5) and $V_{i,j}$ values obtained from Table 5.3.2-2 [8] based on the entry's row and column indexes. Table 2 shows calculations of the corresponding values of a few entries of H_{b1}. The constructed H_{b1} has dimensions of 46×68.

Table 2. Calculation of the entry values of H_{b1}.

Entry of H_{b1}	$V_{i,j}$	$mod\ (V_{i,j}, z_c)$	Corresponding Values
$h_{0,0}$	$V_{0,0} = 223$	$mod\ (223, 56)$	55
$h_{0,1}$	$V_{0,1} = 16$	$mod\ (16, 56)$	16
$h_{0,2}$	$V_{0,2} = 94$	$mod\ (94, 56)$	38
$h_{0,3}$	$V_{0,3} = 91$	$mod\ (91, 56)$	35
$h_{0,4}$	$V_{0,4} = -1$	$mod\ (-1, 56)$	-1
$h_{0,5}$	$V_{0,5} = 74$	$mod\ (74, 56)$	18
$h_{0,6}$	$V_{0,6} = 10$	$mod\ (10, 56)$	10
$h_{0,7}$	$V_{0,7} = -1$	$mod\ (-1, 56)$	-1
$h_{0,8}$	$V_{0,8} = -1$	$mod\ (-1, 56)$	-1
$h_{0,9}$	$V_{0,9} = 0$	$mod\ (0, 56)$	0

In order to achieve the desired information lengths and rate adaptation in 5G NR QC-LDPC codes, a process of shortening and puncturing is carried out. The characteristics of BGM1 are described in the Table 3. The BGM H_{b1} is shown in Table 4.

Table 3. Parameters of BGM1.

Characteristics	**BGM1 (H_{b1})**
Number of block columns (n_b)	68
Number of block rows (m_b)	46
Number of edges	316
Column weights (w_v)	1 to 30
Row weights (w_c)	3 to 19
Base code rate	1/3

Table 4. Base graph matrix.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	……	67
0	55	16	38	35	−1	18	10	−1	−1	0	37	48	21	47	−1	14	14	−1	29	30	48	25	1	0	−1	−1	−1	……	−1
1	29	−1	45	39	46	7	−1	45	21	31	−1	38	37	−1	23	9	6	26	−1	31	−1	19	0	0	0	−1	−1	……	−1
2	39	35	31	−1	8	12	18	39	41	9	14	−1	−1	21	46	21	−1	30	5	55	34	−1	−1	−1	0	0	−1	……	−1
3	33	18	−1	53	5	−1	45	30	16	−1	34	43	45	35	13	−1	40	18	43	−1	30	46	1	−1	−1	0	−1	……	−1
4	2	10	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	……	−1
5	52	3	−1	30	−1	−1	−1	−1	−1	−1	−1	−1	24	−1	−1	−1	14	−1	−1	−1	−1	18	41	−1	−1	−1	−1	……	−1
6	46	−1	−1	−1	−1	−1	7	−1	−1	−1	−1	21	−1	7	−1	−1	−1	51	24	−1	4	−1	−1	−1	−1	−1	−1	……	−1
7	17	20	−1	−1	48	−1	−1	44	38	−1	−1	−1	−1	−1	46	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
8	33	39	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	−1	−1	36	−1	−1	39	−1	2	44	−1	33	−1	−1	……	−1
9	9	37	−1	−1	−1	−1	−1	−1	−1	−1	45	49	−1	33	−1	−1	−1	17	53	−1	50	−1	−1	−1	−1	−1	−1	……	−1
10	−1	26	53	−1	6	−1	−1	19	26	−1	−1	−1	−1	−1	47	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
11	52	11	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	2	−1	−1	−1	35	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
12	30	7	−1	−1	−1	−1	−1	−1	−1	−1	24	3	−1	28	−1	−1	−1	−1	14	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
13	25	−1	−1	0	−1	−1	−1	16	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	−1	22	−1	−1	−1	……	−1
14	14	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	7	−1	−1	43	23	51	−1	−1	−1	43	−1	−1	−1	−1	−1	……	−1
15	34	8	−1	−1	−1	−1	−1	−1	−1	−1	19	−1	−1	41	−1	−1	−1	−1	41	−1	−1	−1	−1	−1	−1	25	−1	……	−1
16	−1	42	−1	52	−1	−1	−1	−1	−1	−1	−1	43	−1	−1	−1	−1	−1	−1	−1	−1	21	−1	45	−1	−1	−1	−1	……	−1
17	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	54	−1	32	7	−1	−1	−1	4	−1	−1	−1	−1	−1	……	−1
18	−1	31	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	54	32	−1	−1	−1	−1	31	18	−1	−1	−1	−1	−1	−1	……	−1
19	8	6	−1	−1	−1	−1	−1	47	30	−1	8	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
20	49	−1	−1	42	−1	−1	−1	−1	−1	9	−1	46	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	15	−1	−1	−1	−1	……	−1
21	−1	24	−1	−1	−1	19	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	52	−1	−1	−1	50	50	−1	−1	−1	−1	−1	……	−1
22	53	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	3	−1	−1	−1	36	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
23	−1	32	35	−1	−1	−1	−1	−1	−1	−1	0	−1	−1	−1	−1	−1	−1	−1	10	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
24	49	−1	−1	45	8	−1	−1	−1	−1	−1	−1	25	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	12	−1	−1	−1	−1	……	−1
25	−1	1	−1	−1	−1	−1	54	9	−1	−1	−1	−1	−1	−1	25	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
26	51	−1	8	−1	44	−1	−1	−1	−1	15	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
27	−1	40	−1	−1	−1	−1	29	−1	6	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
28	34	−1	−1	−1	41	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	2	−1	−1	−1	−1	−1	……	−1
29	−1	38	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	11	−1	−1	−1	53	−1	−1	2	−1	−1	−1	12	−1	……	−1
30	34	−1	−1	−1	−1	−1	−1	−1	−1	−1	18	−1	−1	42	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
31	−1	7	−1	−1	−1	−1	−1	49	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	9	−1	−1	16	−1	……	−1
32	24	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	41	−1	2	−1	−1	−1	−1	−1	−1	−1	−1	−1	30	−1	−1	……	−1
33	−1	2	49	−1	−1	−1	−1	−1	−1	−1	−1	49	−1	−1	−1	−1	−1	−1	−1	−1	−1	25	−1	−1	−1	−1	−1	……	−1
34	26	−1	−1	−1	−1	−1	−1	18	−1	−1	−1	−1	−1	−1	−1	12	−1	38	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
35	−1	24	−1	−1	−1	−1	5	−1	−1	−1	−1	−1	26	−1	−1	−1	−1	−1	−1	−1	−1	−1	19	−1	−1	−1	−1	……	−1
36	54	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	45	0	−1	−1	6	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
37	−1	25	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	27	−1	−1	−1	−1	−1	−1	−1	−1	−1	26	−1	−1	−1	……	−1
38	11	−1	−1	−1	−1	−1	−1	−1	−1	34	17	−1	10	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
39	−1	12	−1	21	−1	−1	−1	49	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	2	−1	−1	−1	−1	−1	−1	−1	……	−1
40	11	−1	−1	−1	−1	−1	−1	−1	45	−1	−1	−1	−1	−1	−1	−1	−1	40	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
41	−1	23	−1	47	−1	−1	−1	−1	−1	4	−1	−1	−1	−1	−1	−1	−1	−1	55	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
42	2	−1	−1	−1	35	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	22	−1	−1	−1	−1	−1	−1	−1	−1	……	−1
43	−1	38	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	22	−1	22	−1	−1	−1	−1	−1	−1	49	−1	……	−1
44	28	−1	−1	−1	−1	−1	−1	4	−1	9	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	12	−1	−1	−1	−1	……	−1
45	−1	16	−1	−1	−1	−1	9	−1	−1	−1	29	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	……	−1

4. Proposed Architecture

The suggested FPGA-based partially parallel stochastic decoder's top-level design is presented in this section. It has runtime flexibility and the capability of decoding received messages corresponding to the set of seven code rates $R = 1/3, 2/5, 1/2, 2/3, 3/4, 5/6$, and $8/9$ and codeword length $N = 3808$ for QC LDPC code compliant to the 5G NR standard. Initially, the architecture determines the maximum matrix dimensions, such as the number of block columns $N_B = 68$, the number of block rows $M_B = 46$, lifting size $Z_c = 56$, column-weight $W_v = 30$, and row-weight $W_c = 19$, for this decoder from the supported code rate set. The architecture comprises various basic modules such as a stochastic variable node decoder (SVND), a stochastic check node decoder (SCND), BGM read-only memory (ROM), a controller unit, an intrinsic message memory unit (IMMU), and a routing network between modules SVND and SCND. These modules of the design are explained in the following subsections.

4.1. Layered Decoding Schedule

In partially parallel architectures, only a few SVNs and SCNs are instantiated simultaneously. Figure 5 shows the proposed architecture that connects the SCND to the SVND. The SCND consists of M D flip-flops, used as memory to store updated SCN messages received from SVND. Here, M is the total number of SCNs of the Tanner graph of the decoded LDPC code. The architecture adopted shuffled iterative decoding [26], in which the total number of block columns of the BGM is subdivided into vertical layers: each layer consists of one block column of BGM or η_v columns of the PCM, where η_v is the SVN parallelism factor; these columns are mapped as SVNs of the Tanner graph of the decoded LDPC code. Before the next vertical layer is processed, the SVNs of each layer are processed in parallel, and the results are used to update the connected SCNs. Using this scheduling, the 3808 columns of the 1/3-rate codeword length $N = 3808$ LDPC code PCM are divided into $N_B = 68$ vertical layers: each layer consists of $\eta_v = 56$ columns. Accordingly, the proposed decoder has an SVND, which comprises $\eta_v = 56$ stochastic variable node processing units (SVNPUs), which function based on the SVN update. These units process a layer of $\eta_v = 56$ columns of the PCM during each clock cycle, with these updated values being written back to SCND immediately. Hence, each decoding cycle (DC) requires $\tau_D = N_B = 68$ clock cycles. Each DC creates one new bit in each stochastic bit sequence.

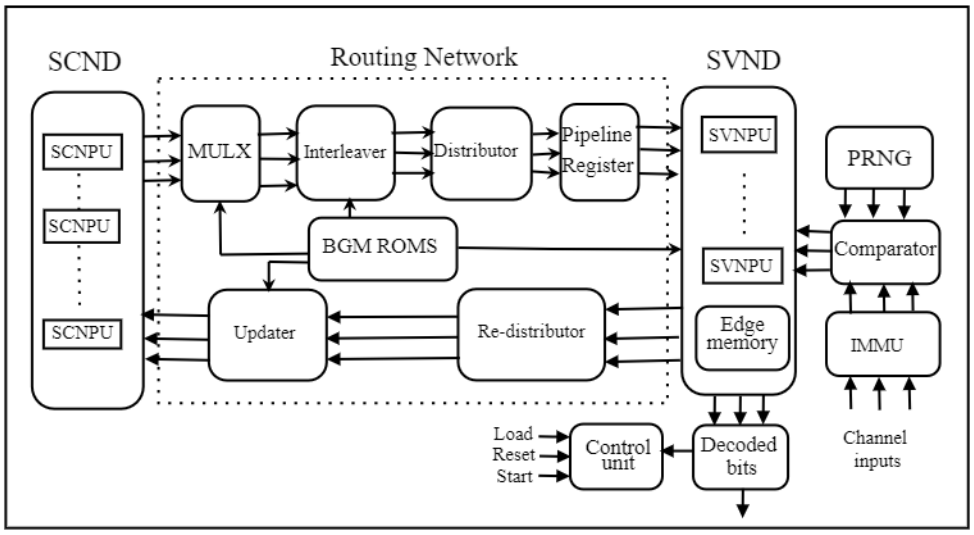


Figure 5. Block diagram of the proposed stochastic decoder.

4.2. BGM ROMs

It is possible to regulate the routing and shifting of intrinsic and extrinsic messages between the routing network, SVNPUs, and SCNPUs by employing a set of ROM blocks that hold the location and shift values of non-‘-1’ entries of each BGM in a hardware-optimized form: the three ROMs, namely ROM1-location, ROM2-shift, and ROM3-weight. Firstly, the row indices of all non-‘-1’ entries within each block column of BGM are represented by the values in the ROM1-location. Second, ROM2-shift saves the shift value difference between each non-‘-1’ entry in each block column and the previous non-‘-1’ entry in the same block row. This value can be determined at design time and saved as described. Lastly, each block column of the BGM’s weight is stored in ROM3-weight. For the proposed architecture, all three ROMs contain $N_b = 68$ locations, each with $W_v = 30$ values: i.e., $N_b \times W_v = 2040$ locations.

4.3. Routing Network

The routing network carries the single-bit SCN-to-SVN messages from the SCND registers to the $\eta_v = 56$ SVNPUs in the SVND; these messages are updated and sent back to the SCND. This routing network combines separate modules called the ’multiplexer’ (MULX), ’interleaver’, ’distributor’, ’re-distributor’, and ’updater’. The functions of these blocks are explained later in this section. The maximum number of SCN-to-SVN message inputs needed for any SVNPU at any time is W_v = 30, which is the maximum number of non-‘-1’ entries in any block column within the allowed code rate set.

4.3.1. Multiplexer

Using the row-index values kept in the ROM1-location, the MULX unit chooses $W_v = 30$ blocks of $Z_c = 56$ bits from the SCND. However, instead of connecting each input to each output, this module requires fewer hardware resources by adopting the method of storing ROM1-location row-index values in ascending order [27]. For instance, the block column of any BGM has weight w_v, such that $w_v \leq W_v$. All of the W_v values of ROM1-location are arranged so that the top w_v block row-indices of the non-‘-1’ entries in that block column are in ascending order, while the remaining $W_v - w_v$ values are empty data. The multiplexer designed to support the seven LDPC BGMs of the allowed code rate set with maximum column weight $W_v = 30$ requires 210 connections. By using this method [27], the number of connections decreased to 39% of those previously required.

4.3.2. Interleaver

The $W_v = 30$ MULX output blocks from the SCND are cyclically shifted by $W_v = 30$ parallel barrel shifters in the interleaver module based on shift values stored in ROM2-shift. Before being read by the SVND, a group of $\eta_v = 56$ messages of the sub-matrix must be converted from a row-centric to a column-centric representation based on the corresponding shift value in BGM H_{b1} by a barrel shifter. Each barrel shifter has $Z_c = 56$ inputs and outputs because the shift value can be any integer between 0 and $Z_c = 56$. The hardware description language (HDL) used for the BSs integrated with the proposed flexible decoder architecture is customised for the supported BGMs during the design phase. The choice of multiplexer input for every BGM is influenced by the expansion factor $Z_c = 56$ value linked to the corresponding BGM based on the cyclic shift decomposition algorithm mentioned in [28].

4.3.3. Distributor and Re-Distributor

The distributor takes W_v = 30 blocks of $\eta_v = 56$ bits from SCND and performs a rearrangement process to form $\eta_v = 56$ blocks of W_v = 30 bits. The pipeline registers may optionally latch these blocks before the SVND processes them. After undergoing SVND processing, the resulting $\eta_v = 56$ blocks of W_v = 30 bits are again subjected to a similar rearrangement process by the re-distributor to form W_v = 30 blocks of $\eta_v = 56$ bits.

4.3.4. Pipeline Registers

Stochastic layered decoding involves binary operations, causing each layer’s message updates to either alter or leave prior layer updates unchanged. The decoding schedule needs to store updates from each previous layer before moving on to the next. Adding a single pipeline stage to the suggested architecture can sometimes improve BER performance without negatively impacting the decoding process. In some BGMs, a column-wise permutation occurs, allowing for column-orthogonality in block columns. Adding a pipeline stage during the data path does not negatively impact decoding performance when processed in this order [27]. If a permutation is not found, introducing a few τ vacant stall layers among non-orthogonal columns can be developed. Finding the necessary minimum values of τ for each BGM and adding this supplementary pipeline stage is advantageous. Decreasing the number of block rows while keeping the number of block columns constant can enhance the coding rate. Simulations show that high-rate codes require more τ vacant stall layers among non-orthogonal columns compared to low-rate codes.

4.3.5. Updater

Using the values from the ROM1-location, this component transfers these W_v = 30 blocks of $\eta_v = 56$ bits back into the SCND at the proper location. Moreover, the ROM3-weight is utilized in this case to guarantee that only the most recent W_v values are written, removing any potential interference from non-'-1' entries or "don't care" data. As D-flip-flops are being used to implement the SCND, each layer can be written right away.

4.4. Stochastic Variable Node Decoder (SVND)

The SVND module contains $\eta_v = 56$ parallel stochastic variable node processing units (SVNPUs). These SVNPUs can process one block column of the BGM H_{b1}. Hence, each SVNPU processes one column of the PCM. Each SVNPU has multiple inputs and outputs equal to $W_v + 1$, where W_v is the highest column weight in the supported BGM code rate set. The extra input and output are utilized, respectively, for channel-intrinsic messages and the estimation of the decoded bit. It is essential to consider that the largest number of inputs and outputs must be taken into account to guarantee that each SVNPU can decode any arbitrary column-weight column in the code rate set. The SVND accepts the maximum SVN weight $W_v = 30$ input bits from the connected SCNs and an intrinsic bit from the intrinsic message memory unit (IMMU). By considering such values, SVND implements the SVN update operation, and the updated extrinsic values are sent to the corresponding SCNPUs.

Stochastic Variable Node Processing Unit (SVNPU)

In this stochastic architecture, the chance of SVNPU output being in the hold state increases as inputs to SVNPU increase [24]. The SVNPU forces the current output to continuously repeat the previous output over a certain period when the two input bits of the SVNPU are not equal, known as the hold state of the SVNPU [25]. Due to this state, the decoder is unable to make correct decisions. It interrupts the decoder convergence, and its bit error rate (BER) performance degrades. Hence, the architecture of high-column-weight SVNPU is built using low-weight sub-nodes with memory blocks; typically, $w_v \leq 4$ sub-nodes are employed.

As shown in Figure 6 [24], the column-weight $w_v = 6$ SVNPU is constructed using two column-weight $w_v = 3$ and one $w_v = 2$ sub-nodes. The memory blocks are internal memory (IM), which has a length of $I = 2$ bits for each sub-node, and edge memory (EM), which has a length of $E = 64$ bits at the output edge. The critical role of EM and IM is to minimize the correlation produced by the hold state in a stochastic sequence and disrupt the correlation of the stochastic sequence by randomizing stochastic bits [29]. We used a dual-tree design [27] to build a high-column-weight SVNPU that would be flexible enough to handle any active inputs and outputs up to the maximum $\lambda = W_v + 1$, which is a power of two. For instance, column weight $W_v = 30$ of SVNPU has $\lambda = 32$ inputs and outputs. As a result, the adopted SVNPU dual-tree design requires a total of

$S = 3 \times \lambda - 6 = 90$ sub-nodes. Two significant components, summing and combining, each with $t = \log_2(\lambda) - 1 = 4$ stages, make up the dual-tree structure.

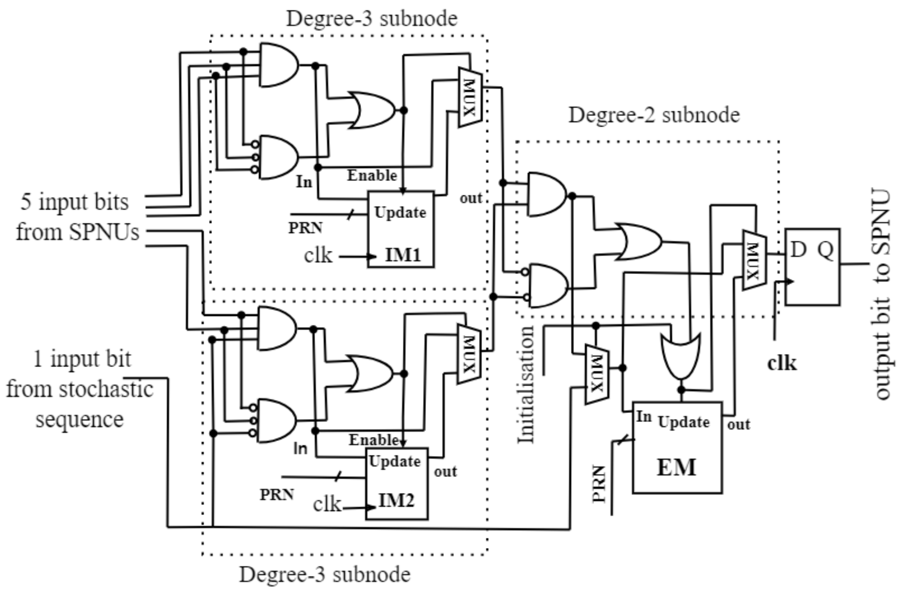


Figure 6. Block diagram of SVNPU with high $w_v = 6$.

4.5. Control Unit

The control unit of the decoder manages internal and external control signals, including commands to stop or proceed with the iterative decoding. A new group of $\eta_v \times w$-bit intrinsic probabilities may need to be loaded into the IMMU, and this can be indicated by the LOAD signal. The decoder must decode a new frame, and all modules must be reset to their default conditions using the RESET signal. The ability of the proposed architecture to swap the current BGM to decode the message contained in the IMMU during a single clock cycle is one of its essential characteristics. The supporting BGMs are parameterised in ROMs at compile-time to enable this flexibility level. The process of decoding is started and maintained by the START signal. As long as this signal is active, the decoder keeps doing iterative decoding; the decoding operation is stopped once if the START signal is absent.

5. Design Flow

For instance, a QC LDPC code with a codeword length of N = 3808 and a code rate of R = 1/3 that complies with the 5G NR standard has a BGM H_b with the dimensions 46×68. This BGM has 308 non-'-1' entry values, which is equal to 68×4.52, and an average variable node degree of 4.52. The BGM H_b forms the basis for the representation and construction of the PCM H. Therefore, the calculation for non-zero entries in the PCM is determined by multiplying 308 by 56, which equals 17,248, where Z_c = 56 is the lifting size or expansion factor. Consequently, this results in twice the 17,248 interconnections that exist between the variable and check nodes of the decoder, which makes the structure extremely complicated. Modelling these thousands of interconnections in hardware description language (HDL) takes at least thirty minutes. This modelling is extremely time-consuming when dealing with a partially parallel architecture. The process of simulating and debugging the hand-coded HDL model is both extremely challenging and time-consuming, respectively. The process of modelling the decoder with a different code length turns out to be an activity that is both unproductive and repetitive. Because of this, automating the design flow to remove repetitive and redundant tasks requires an efficient design methodology. The above-mentioned problems were solved by creating an automation tool in MATLAB using the C++ programming language; the tool takes less than a minute to generate connections between the nodes of the proposed decoder.

This section outlines the suggested design flow, which enables the automated flexibility in design and creates an optimal FPGA-based stochastic LDPC decoder for a selected set of 5G QC LDPC codes. The flowchart in Figure 7 depicts the design flow. In the first step, the construction of the required code rate BGM set is based on the user inputs like message length, code rate, and lifting sizes of the 5G LDPC code. The second step has two tasks: The first task entails determining the decoder's parameters, such as the maximum of the matrix dimensions and weights, namely N_b, M_b, Z_c, W_v, and W_c, which describe the chosen set of supported BGMs. Based on lifting size Z_c, it is possible to determine the parallelism factor η_v from these. For the $N = 3808$ LDPC code rate set, the maximum values are $N_b = 68$, $M_b = 46$, $Z_c = 56$, $W_v = 30$, and $W_c = 19$. Another task is to extract the positions and shift values of the non-'-1' entries in each BGM and arrange them consistent with the ROMs. Considering the parameters derived in the earlier step, the design flow utilizes a high-level synthesis (HLS) tool [30] to produce the hardware description language (HDL) SystemVerilog code for the suggested architecture written in the high-level language C++. After register transfer level (RTL) modelling of the decoder, it is synthesized using the Xilinx synthesis tool to measure the hardware requirements of the decoder. The bit error rate (BER) simulations have been implemented on the FPGA test setup.

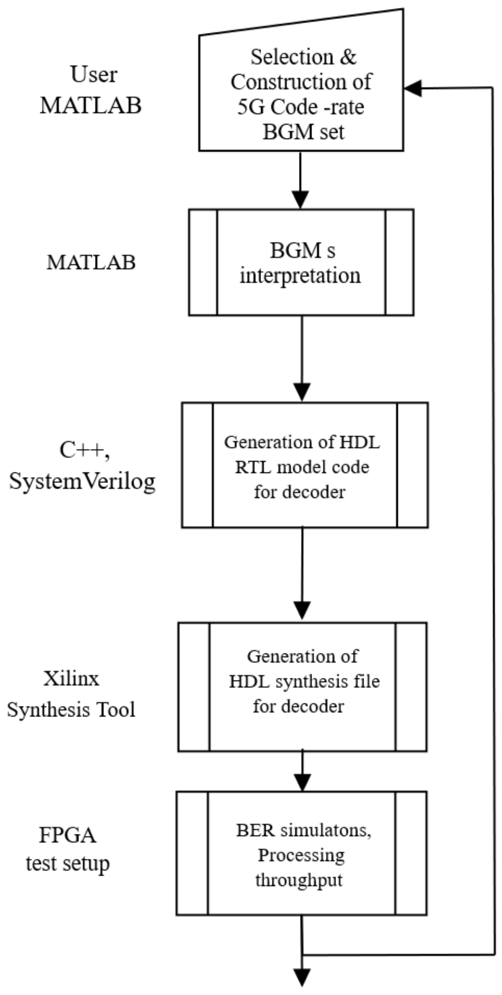

User MATLAB
Selection & Construction of 5G Code -rate BGM set
MATLAB
BGM s interpretation
C++, SystemVerilog
Generation of HDL RTL model code for decoder
Xilinx Synthesis Tool
Generation of HDL synthesis file for decoder
FPGA test setup
BER simulatons, Processing throughput

Figure 7. Flowchart for offline design and implementation of the proposed decoder.

6. Implementation Results and Discussion

6.1. Approach

After the offline design method is completed, the HDL code description of the decoder for the required code rate set is generated, which is further synthesized and integrated as the HDL code on the Xilinx Kintex-7 XC7K160T-series FPGA board [31] to determine the hardware resource utilisation and processing throughput. An additional parameter that is measured is the transmission energy efficiency of the synthesized decoder in terms of the channel's signal-to-noise power ratio per bit E_b/N_o at a required BER of 10^{-6} for each target BGM. Simulations are carried out to evaluate the transmission energy efficiency of the synthesized decoder. The simulations entail a minimum of 100 frame errors per BER measurement and a maximum of 600 DCs per frame.

The intrinsic channel LLRs serve as input to the decoder module of the FPGA board, which is received via the RS232 port of the computer. In order to facilitate communication with the computer, an RS232 transceiver module has been designed. The MATLAB environment in the computer is utilised to send and receive the same set of decoded LLRs after completing the decoding process on the FPGA. Subsequently, MATLAB compares the input and output LLRs of the FPGA and estimates the BER performance.

6.2. Results

In order to analyse the performance of the proposed decoder, we consider three parameters: BER performance, hardware utilisation, and processing throughput.

6.2.1. BER Performance

Figure 8 shows BER performance of the proposed decoder, which delivers BER = 10^{-6} at 2.65 dB of E_b/N_o for a block length of 3808 with a base code rate of 1/3. It has been observed that SD provides better error correction performance compared with the conventional sum–product (SPA) and min–sum (MS) algorithms. Notably, decoding iterations of SD require more clock cycles than conventional designs. Additionally, it has been observed that applying noise-dependent scaling of 0.86 [29] to the received channel probabilities improves performance for lower code rates such as 1/3 and 2/5. Conversely, this performance declines for higher code rates such as 5/6 and 8/9.

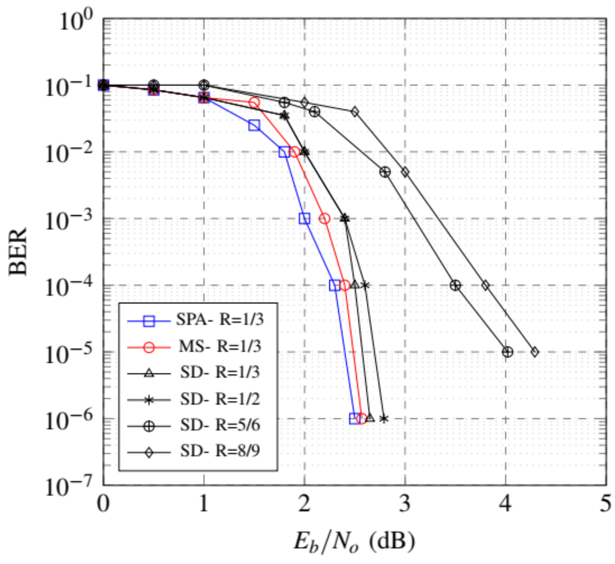


Figure 8. BER plot of various algorithms and code rates for N = 3808.

6.2.2. Hardware Utilisation

The proposed design's advantage lies in the significant reduction in decoder complexity. Table 5 compares the suggested design and the min–sum-based decoder architectures, indicating that the former requires less hardware by approximately 37%. A crucial parameter that affects the decoder's area and routing complexity is the number of interconnecting wires between its nodes. In this regard, the suggested design outperforms the min–sum-based decoder by requiring 34,729 fewer interconnect wires. Other parameters of the suggested design are comparable to those of the min-sum-based decoder. The results in Table 6 indicate that the increase in coding rate significantly impacts the SD decoder's decoded processing throughput. However, at the same time, it negatively affects the error correction performance.

Table 5. FPGA implementation results.

Standard	**5G**	**5G**
Code length	3808	3808
Base code rate	1/3	1/3
Sub-matrix size	56	56
Implementation	Kintex-7 FPGA	Kintex-7 FPGA
Decoding algorithm	Stochastic decoding	Min–Sum
Scheduling	Column-layered	Row-layered
No. of interconnects	34,729	138,916
Intrinsic message width	8-bit	4-bit
Extrinsic message width	1-bit serial	4-bit
LUTs	8278	12,962
Slice registers	1767	2041
DCs or Itrs	≈620 DCs	15
Avg. throughput	≈953 Mbps	1.5 Gbps
E_b/N_o at BER = 10^{-6}	2.65 dB	2.57 dB

Table 6. SD results of various code rates for $N = 3808$.

Active Code-Rate	**No. of Clock Cycles per DC**	**LUTs (k)**	**Slice Registers (k)**	**Throughput (Mbps)**	**E_b/N_o at BER = 10^{-6}**	**No. of DC per Frame**
1/3	68	8.2	1.7	953.4	2.65 dB	620
2/5	68	8.2	1.7	964.3	2.69 dB	530
1/2	68	8.2	1.9	1100.9	2.79 dB	450
2/3	68	8.2	1.9	1189.3	3.28 dB	430
3/4	68	8.2	1.9	1240.5	3.87 dB	400
5/6	68	8.2	1.9	1267.6	4.02 dB	360
8/9	68	8.2	1.9	1298.2	4.29 dB	330

6.2.3. Processing Throughput

The proposed architecture requires a reduction in hardware resources, although it performs with a considerably lower processing throughput expense. This lower throughput of stochastic decoders is due to the high number of decoding cycles (DCs) needed to reach a correct code word. Due to the high number of DCs and the partially parallel decoder architecture, the problem is made worse by the fact that each DC requires many clock cycles, which reduces decoding throughput. The SD architecture requires a maximum of 620 DCs per frame, and each decoding cycle necessitates 68 clock cycles for code length $N = 3808$. Table 5 compares both designs and concludes that the processing throughput of the SD design is about 38% lower than that of the min-sum design. Table 6 demonstrates that an increase in the code rate results in a concurrent increase in the processing throughput.

6.3. Comparative Analysis

To facilitate a comparative analysis, the Verilog hardware description language (HDL) was employed to model the architecture. It was subsequently subjected to simulation to verify its functionality using a test pattern generated by a C++ simulator. The design functions were verified successfully, following which, the architecture was synthesized while adhering to suitable time and area constraints. Synopsys design tools were employed during both the simulation and synthesis stages in combination with TSMC 65 nm CMOS standard cell technology. The results obtained post-synthesis are presented in Table 7. Specifically, the proposed SD architecture occupies an area of 1.10 mm^2 and achieves a throughput of 1.12 Gbps, while the power consumption is 410 mW.

Table 7. Comparative results.

Design	**Proposed**	**[32]**	**[33]**	**[34]**	**[35]**
Standard	5G-NR	5G-NR	802.16e	802.15.3c	802.11n
Code length	3808	3808	2304	672	1944
Base code rate	1/3	1/3	1/2	1/2	1/2
Decoding algorithm	SD	CMS	NMS	NMS	MS
Scheduling	Column-layered	Row-layered	Row- layered	Row-layered	Row-layered
Extrinsic message width	1-bit	4-bit	6-bit	4-bit	4-bit
Sub-matrix size	56	56	96	21	81
DCs or Itrs	620	10	10	5	10
Area (mm^2)	1.10	1.49	2.9	2.25	4.88
Throughput (Gbps)	1.12	3.04	2.20	5.28	4.5
Power (mW)	410	259	870	182	523

The comparison of the proposed SD with other LDPC decoders is provided in Table 7 considering implementation and performance. The proposed design has been shown to reduce the decoder architecture's complexity significantly. These designs exhibit varying implementation parameters and include code length and decoding algorithms such as combined min–sum (CMS) and normalised min-sum (NMS). Compared to the reported decoders, the proposed architecture exhibits area efficiency increases of 26% compared to [32], 62% compared to [33], 51.11% compared to [34], and 77.46% compared to [35], while delivering energy efficiency improvements of 52% compared to [33] and 21.6% compared to [35].

7. Conclusions

In this paper, we constructed a BGM of QC LDPC code meeting the 5G NR standard. We presented stochastic decoding to LDPC codes as a hardware-efficient alternative to SPA- and MS-based LDPC decoders. From simulations, we observed that with the inclusion of features like scaling and edge memory, stochastic decoding performs almost at the level of SPA-based LDPC decoder in terms of BER performance. Compared to the reported decoders, the proposed architecture exhibits area efficiency improvements of 26% compared to the CMS-based decoder, 62% compared to the NMS-based decoder, and 77.46% compared to the MS-based decoder, while delivering energy efficiency improvements of 52% compared to the NMS-based decoder and by 21.6% compared to the MS-based decoder.

Author Contributions: S.P.T. conceptualized the idea of this research, conducted experiments, collected data, and prepared the original version. R.A. reviewed, analyzed data, and updated the manuscript. R.P. supervised, validated, reviewed. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used for the experiments reported in this paper are available upon request from the authors via email.

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

References

1. Gallager, R.G. Low-density parity-check codes. *IRE Trans. Inf. Theory* **1962**, *8*, 21–28. [CrossRef]
2. ETSI. *ETSI EN 302 307 v1.3.1 Digital Video Broadcasting (DVB)*; Second Generation; ETSI: Sophia Antipolis, France, 2013.
3. *IEEE 802.11n-2009*; Standard for Information technology-Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE: New York, NY, USA, 2009.
4. *IEEE 802.16-2004*; Standard for Local and Metropolitan Area Networks—Part 16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE: New York, NY, USA, 2004.
5. 3GPP. R1-1611081 Final Report. In Proceedings of the 3GPP TSG RAN WG1 Meeting 86bis, Lisbon, Portugal, 10–14 October 2016; pp. 83–89.
6. MacKay, D.J.C.; Neal, R.M. Near Shannon limit performance of low-density parity-check codes. *Electron. Lett.* **1996**, *32*, 1645–1646. [CrossRef]
7. Chung, S.Y.; Forney, G.D.; Richardson, T.J.; Urbanke, R. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Commun. Lett.* **2001**, *5*, 58–60. [CrossRef]
8. Ad-Hoc Chair (Nokia). Chairman's Notes of Agenda Item 7.1.4. Channel Coding. 3GPP TSG RAN WG1 Meeting AH 2, R1-1711982. 2017. Available online: https://portal.3gpp.org (accessed on 9 December 2022).
9. Fossorier, M.P. Quasi cyclic low-density parity-check codes from circulant permutation matrices. *IEEE Trans. Inf. Theory* **2004**, *50*, 1788–1793. [CrossRef]
10. Tanner, R. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory* **1981**, *27*, 533–547. [CrossRef]
11. Wiberg, N. *Codes and Decoding on General Graphs*; Citeseer: Princeton, NJ, USA, 1996.
12. Hailes, P.; Xu, L.; Maunder, R.G.; Al-Hashimi, B.M.; Hanzo, L. A Flexible FPGA-Based Quasi-Cyclic LDPC Decoder. *IEEE Access* **2017**, *5*, 20965–20984. [CrossRef]
13. Zhang, C.; Wang, Z.; Sha, J.; Li, L.; Lin, J. Flexible LDPC Decoder Design for Multigigabit-per-Second Applications. *IEEE Trans. Circuits Syst. Regul. Pap.* **2010**, *57*, 116–124. [CrossRef]
14. Kschischang, F.R.; Frey, B.J.; Loeliger, H.A. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory* **2001**, *47*, 498–519. [CrossRef]
15. Angarita, F.; Valls, J.; Almenar, V.; Torres, V. Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor. *IEEE Trans. Circuits Syst. Regul. Pap.* **2014**, *61*, 2150–2158. [CrossRef]
16. Gaudet, V.C.; Rapley, A.C. Iterative decoding using stochastic computation. *Electron. Lett.* **2003**, *39*, 299–301. [CrossRef]
17. Hayes, J.P. Introduction to Stochastic Computing and its Challenges. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 7–11 June 2015; pp. 2–4.
18. Lee, X.R.; Chen, C.L.; Chang, H.C.; Lee, C.Y. A 7.92 Gb/s 437.2 mW Stochastic LDPC Decoder Chip for IEEE 802.15.3c Applications. *IEEE Trans. Circuits Syst. Regul. Pap.* **2015**, *62*, 507–516. [CrossRef]
19. The IEEE 802.15.3c Working Group Std. (WPAN). [Online]. Available online: https://www.ieee802.org (accessed on 1 November 2009).
20. Chandrasetty, V.A.; Aziz, S.M. *Resource Efficient LDPC Decoders: From Algorithms to Hardware Architectures*; Academic Press: Cambridge, MA, USA, 2017.
21. Gross, W.J.; Gaudet, V.C.; Milner, A. Stochastic implementation of LDPC decoders. In Proceedings of the Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 30 October–2 November 2005; pp. 713–717.
22. Zhang, D.; Li, H. A Stochastic-Based FPGA Controller for an Induction Motor Drive with Integrated Neural Network Algorithms. *IEEE Trans. Ind. Electron.* **2008**, *55*, 551–561. [CrossRef]
23. Dinu, A.; Cirstea, M.N.; McCormick, M. Stochastic implementation of motor controllers. In Proceedings of the 2002 IEEE International Symposium on Industrial Electronics, L'Aquila, Italy, 8–11 July 2002; Volume 2, pp. 639–644.
24. Tehrani, S.S.; Mannor, S.; Gross, W.J. Fully Parallel Stochastic LDPC decoders. *IEEE Trans. Signal Process.* **2008**, *56*, 5692–5703. [CrossRef]
25. Winstead, C.; Gaudet, V.C.; Rapley, A.; Schlegel, C. Stochastic iterative decoders. In Proceedings of the International Symposium on Information Theory, ISIT 2005, Adelaide, SA, Australia, 4–9 September 2005; pp. 1116–1120.
26. Zhang, J.; Fossorier, M.P.C. Shuffled iterative decoding. *IEEE Trans. Commun.* **2005**, *53*, 209–213. [CrossRef]
27. Hailes, P. Design and Implementation of Flexible FPGA-Based LDPC Decoders. Ph.D. Thesis, University of Southampton, Southampton, Malaysia, 2018.
28. Jung, Y.; Jung, Y.; Lee, S.; Kim, J. Low-complexity multi-way and re-configurable cyclic shift network of QC-LDPC decoder for Wi-Fi/WIMAX applications. *IEEE Trans. Consum. Electron.* **2013**, *59*, 467–475. [CrossRef]
29. Tehrani, S.S.; Gross, W.J.; Mannor, S. Stochastic decoding of LDPC codes. *IEEE Commun. Lett.* **2006**, *10*, 716–718. [CrossRef]

30. Nane, R.; Sima, V.M.; Pilato, C.; Choi, J.; Fort, B.; Canis, A.; Chen, Y.T.; Hsiao, H.; Brown, S.; Ferrandi, F.; et al. A Survey and Evaluation of FPGA High-Level Synthesis Tools. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2016**, *35*, 1591–1604. [CrossRef]
31. Xilinx. *7 Series FPGA Configurable Logic Block User Guide—UG474*; Xilinx: San Jose, CA, USA, 2014.
32. Thi Bao Nguyen, T.; Nguyen Tan, T.; Lee, H. Low-complexity high-throughput QC-LDPC decoder for 5G new radio wireless communication. *Electronics* **2021**, *10*, 516. [CrossRef]
33. 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]
34. Li, M.-R.; Yang, C.-H.; Ueng, Y.-L. A 5.28-Gb/s LDPC decoder with time-domain signal processing for IEEE 802.15. 3c applications. *IEEE J. Solid State Circuits* **2016**, *52*, 592–604. [CrossRef]
35. Tsatsaragkos, I.; Paliouras, V. A reconfigurable LDPC decoder optimized for 802.11 n/ac applications. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *26*, 182–195. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.