electronics

MDPI

Article

Efficient Layered Parallel Architecture and Application for Large Matrix LDPC Decoder

Jimin Wang †, Jiarui Yang, Guojie Zhang, Xiaoyang Zeng and Yun Chen *,†

State Key Laboratory of Integrated Chip and System, Department of Microelectronics, Fudan University, Room 220, 825 Road Zhangheng, Shanghai 201203, China

* Correspondence: chenyun@fudan.edu.cn

† These authors contributed equally to this work.

Abstract: For a 50G passive optical network (PON) low-density parity-check (LDPC) decoder, decoding performance and area efficiency must be balanced. This paper adopts a layered decoder method to improve the area efficiency of the decoder. By parallel processing of three submatrices and storage reuse of node information, optimizing the matrix partitioning and processing order of the 50G-PON standard, the throughput of 1235 bps was reached under 100 MHz circuit frequency in field-programmable gate array (FPGA) implementation, and 9.864 Gbps was achieved based on Taiwan semiconductor manufacturing company (TSMC) 65 nm synthesis with 800 MHz circuit frequency in an area of 2.61 mm^2 by proposing a mechanism of spare decision storage to avoid errors caused by quantization overflow of the decoder and using full verification to terminate decoding in advance to improve performance. Finally, at an input bit error rate (BER) of 2.3×10^{-2} (signal-to-noise ratio (SNR) of about 3.72 dB), the output BER was lower than 10^{-12}, and the throughput area rate (TAR) also increased by 2 to 4 times compared with other papers. In conclusion, an area-efficient LDPC decoder without sacrificing decoding performance is made.

Keywords: LDPC; decoder; 50G-PON

check for updates

Citation: Wang, J.; Yang, J.; Zhang, G.; Zeng, X.; Chen, Y. Efficient Layered Parallel Architecture and Application for Large Matrix LDPC Decoder. *Electronics* **2023**, *12*, 3784. https://doi.org/10.3390/electronics12183784

Academic Editor: David Defour

Received: 16 July 2023
Revised: 25 August 2023
Accepted: 27 August 2023
Published: 7 September 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 (LDPC) code was first proposed by Gallager in his PhD thesis in 1962 [1]. Since Mackay discovered the decoding performance of LDPC code close to Shannon's limit in 1997 [2], various papers and research on LDPC codes have come out one after another, and they have also been applied commercially according to various standards. Currently, LDPC code has been used as a channel error correction code in communication standards such as WiMax, 10GBASE-T and 5G. Since the decoder module in a channel codec is the key module that affects the data transmission rate, it is generally inclined to adopt the design method of improving performance, which needs to use a large number of logic computing devices and memory devices, and its hardware resource consumption is often large. Obviously, the channel codec module occupies a large part of the power consumption and area in the whole communication baseband processing chip, so the area power consumption performance of the designed channel codec module will directly affect the baseband chip area power consumption. Naturally, it is very important to study the technical means and design methods of high-performance decoders to improve area efficiency and power efficiency.

In 1999, Mackay et al. proposed a sum-product algorithm that can handle Gallager codes and MN (Mackay–Neal) codes. The sum-product algorithm (SPA) [3], also known as the two-phase message passing (TPMP) algorithm, is actually a probabilistic domain belief-propagation (BP) algorithm. In order to reduce the amount of operations when decoding, the SPA algorithm in the logarithmic domain [4] emerged, which enables the replacement of multiplication with addition. Subsequently, the min-sum algorithm (MSA) [5,6] was

proposed by further approximating the check notes (CN) computation with the minimum of the set of input nodes. Further, the offset min-sum algorithm (OMS) [7] and the normalized min-sum algorithm (NMS) [8] were introduced to compensate for the loss of decoding performance of MSA.

In terms of decoder hardware design, the turbo-like decoding message passing (TDMP) algorithm proposed by Mansour in 2003 significantly increased the iterative decoding speed of the decoder, allowing the corresponding hardware implementation to increase the throughput without sacrificing the decoding performance [9]. TDMP is actually a typical horizontal layered algorithm [10,11], which divides the check nodes into several layers, and the update results of the last layer can be utilized in the subsequent layers, thus speeding up the update process. In contrast, J. Zhang published a paper on the vertical-layered algorithm [12], where the checksum matrix is divided vertically. Moreover, it has been shown that both horizontal- and vertical-layered updates can theoretically achieve twice the decoding convergence speed of TPMP [13]. In addition, the residue-based layered algorithm proposed by researchers can further improve decoding convergence speed [14].

For long codes, the traditional LDPC decoder generally adopts the hierarchical update method, which divides the update of all nodes into multiple layers. However, the throughput of the hierarchical decoder is smaller than that of full parallel decoding because of fewer processing units. However, the fully parallel decoder hardware scale will increase with the increase in code length, and it is not suitable for long codes.

In order to improve the throughput and area efficiency of the decoder at the same time, this paper considers updating the two kinds of nodes at the same time and, at the same time, increasing the parallelism of node processing and selecting a partial block parallel scheme. In the partial block parallel decoder, the computation and memory/write need to be increased several times. Too many partition times of the basis matrix will introduce a large number of bubbles, which will cause a great negative impact on the throughput of the decoder. In this way, the condition of six read and write accesses per cycle can be achieved by using double-ended memory and dividing the basis matrix into three blocks. In order to reduce the resource consumption of the check module, this paper presents a serial full-check method by using the cyclic shift characteristic of check matrix. In order to solve the error caused by quantization overflow processing of variable notes(VN) information, a kind of hard-decision memory–backup memory (BM) is proposed in this paper.

The Section 1 of this paper is the introduction, which mainly introduces the development of LDPC decoding technology and the contribution of this paper. The Section 2 is about the materials and methods. Firstly, the decoding algorithm of LDPC is introduced, and then the quantization method and error are discussed. The Section 3 is the design and implementation, showing the hardware block diagram of the decoder and introducing the three techniques used in the decoder. Section 4 shows the implementation results of the decoder, and a summary of them is in Section 5.

2. Materials and Methods

2.1. Decoding Algorithm

LDPC codes are defined by a (M, N)-dimensional parity matrix $\mathbf{H}$. The N columns of the check matrix correspond to the coded code elements, and the M rows correspond to the check equation. For a code word $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$, $\mathbf{x}$ is a valid code word when and only when $\mathbf{H}\mathbf{x}^T = 0$, and the decoding result may correspond to the original encoded code word, thus obtaining the reduced sequence of information code elements.

The meanings and algorithm parameters represented by subsequent symbols are given below: (1) c_n represents the symbol information of the nth code element in the encoded code word; (2) y_n represents received signal; (3) I_n represents the channel information; (4) C_m represents the mth check node; $N(m)\backslash n$ represents the set of VNs connected to the check node C_m, except the variable node V_n; and R_{mn} represents CN information passed from the check node C_m to the variable node V_n; (5) V_n represents the nth variable node; (6) Q_{mn} represents VN information passed from the variable node V_n to the check node

C_m; (7) Posterior probability information: Λ_n represents the pseudoposterior probability information of the nth code word in the computed code word, whose decoding result c_n can be obtained after a hard judgment.

Different from tradition layered algorithms, the update order rearrangement technique used in [15] by Lee et al. does not require the use of FIFO (First in First out). It updates VN information Q_{mn} instead of the posterior probability information Λ_{mn} in the iteration. Its algorithm flow is shown below.

Step 1: Initialization of channel information:

The initial external information of the VN is initialized by the following formula, where σ^2 is the channel noise variance estimate.

$$Q_n = I_n = -\frac{2y_n}{\sigma^2} \tag{1}$$

where the number of iterations $i = 0$, and the initial CN also needs to be initialized.

$$R_{mn} = 0 \tag{2}$$

Step 2: Iteration:

Prepare the subiterations of the row block in the ith iteration; the number of subiterations layers m increases.

Step 3: Row block subiterations:

Computation of the posterior probability information, where $m^{'}$ is the check node label of the most recently updated V_n;

$$\Lambda_n = Q_n + R_{m^{'}n} \tag{3}$$

The current row block VN is updated and written back to memory;

$$Q_n^{'} = \Lambda_n - R_{mn} \tag{4}$$

the current row block CN is updated after searching the array by the minimum value;

$$R_{mn} = \alpha \times \prod_{n^{'} \in N(m) \backslash n} sign\left(Q_{n^{'}}\right) \times \min_{n^{'} \in N(m) \backslash n} \left|Q_{n^{'}}\right| \tag{5}$$

Step 4: Hard decision:

Generate the updated code word, if it passes the check to satisfy $\mathbf{H}\mathbf{x}^T = 0$, then terminate the decoding; otherwise, m plus one and return to the third step for the next layer of subiterations. If all line blocks are updated, then i plus one and return to the second step for the next iteration.

$$c_n = \begin{cases} 0 & \Lambda_n \geq 0 \\ 1 & otherwise \end{cases} \tag{6}$$

Table 1 compares the differences between the normal TDMP and the reordered decoding flow. Because the decoder hardware takes multiple cycles to find the magnitude minimum of VN, the part of calculating CN is highlighted.

The computation of CN in the ordinary decoding flow is between the read and write cycles of the Λ_n being updated; the computation of Λ_n depends on the new CN and Q_{mn}, the intermediate variable Q_{mn} needs to be stored in FIFO. In contrast, the computation of CN after rearrangement is performed after the read and write cycles of the updated Q_n, and there is no dependency, and the updates of VN and CN are performed successively; so, a FIFO can be saved in the case of using single-ended memory to store VNs, which effectively improves the area efficiency of the decoder.

Table 1. Comparison of reordered flow.

	Normal	**Reordered**
mth row	read Λ_n	read Q_n
	calc $Q_{m'n}$	calc Λ_n
	calc $R_{m'n}$	$calcQ_n^{'}$
	calc $\Lambda_n^{'}$	write $Q_n^{'}$
	write $\Lambda_n^{'}$	calc $R_{m'n}$
$m^{'}$th row	read $\Lambda_n^{'}$	read $Q_n^{'}$
	calc $Q_{m'n}$	calc Λ_n
	calc R_{mn}	$calcQ_n^{''}$
	calc $\Lambda_n^{''}$	write $Q_n^{''}$
	write $\Lambda_n^{''}$	calc R_{mn}

2.2. Quantization

Quantization is an important part of circuit hardware design. In addition, the analysis of the impact of quantization on the LDPC algorithm helps to optimize the design and implementation of the decoder circuit.

For value k, quantization is performed as in Equation (7), where k is multiplied by a factor $1/\Delta_k$ for linear mapping, and then the fractional part is discarded to obtain an integer. In addition, since the bit width is finite, the number of symbols it can represent is limited, so it is necessary to limit the amplitude of k, as in Equation (8), where Q_k represents the maximum amplitude that can be represented by quantization. For q bits quantization, the set of values that can be represented by the binary complement is $\{-2^{q-1}, -2^{q-1}+1, \ldots, 2^{q-1}-1\}$; the set of values represented by the original binary code is $\{-2^{q-1}+1, -2^{q-1}+2, \ldots, 2^{q-1}-1\}$. Since the absolute value of CN is needed to find the minimum value, there are two representations in the circuit at the same time. In order to ensure that no error is generated when converting the two representations to each other, the maximum magnitude $Q_k = 2^{q-1}-1$ is taken.

$$Quant(k) = \left\lfloor \frac{k}{\Delta_k} \right\rfloor \tag{7}$$

$$Limit(k^{'}) = \begin{cases} Q_k \times sign\left(k^{'}\right) & \left|k^{'}\right| \geq Q_k \\ k^{'} & \left|k^{'}\right| < Q_k \end{cases} \tag{8}$$

Overflows in the hardware implementation also introduce errors between the algorithm and the hardware. From Equation (4), the VN read from the memory is 6 bits, but the updated VN needs to be represented by 8 bits; so, it needs to go through another overflow processing, and the actual VN update equation in hardware can be obtained as follows:

$$Q_n^{'} = Limit(Q_n + R_{m'n} - R_{mn}) \tag{9}$$

Obviously, in this case, there is a gap between the value obtained in the hardware circuit and the theoretical value of the algorithm, with a theoretical maximum error of up to $22.5 = \alpha \times 2 \times (2^{5-1}-1)$ for $\alpha = 0.75$. When the decoder is in the first and middle stages of the decoding iteration, most of the VN magnitudes and CN information magnitudes in the hardware are relatively small, and the introduced errors are small and almost negligible. However, at the later stages of the decoding iteration, the decoder tends to converge, and most of the VN magnitudes and CN magnitudes are already large, and the introduced errors are not negligible. If the decoder does not successfully decode in a short time, it may lead to a large increase in error bits in the decoding result.

Figure 1 shows how the number of erroneous bits varies with iterations in the decoding process of the layered decoder considering quantization and overflow, where the horizontal

coordinate is the number of subiterations, and the vertical coordinate is the bit error number remaining in hard decision. As the subiterations proceed, the number of erroneous bits in the code word decreases first, and then it remains relatively stable after decreasing to the minimum value, then increasing sharply later. In most cases, the decoder will stop decoding after a valid code word is decoded, but if it fails to end when the error bits are the lowest, a large number of error bits may impact the BER performance of the decoder.

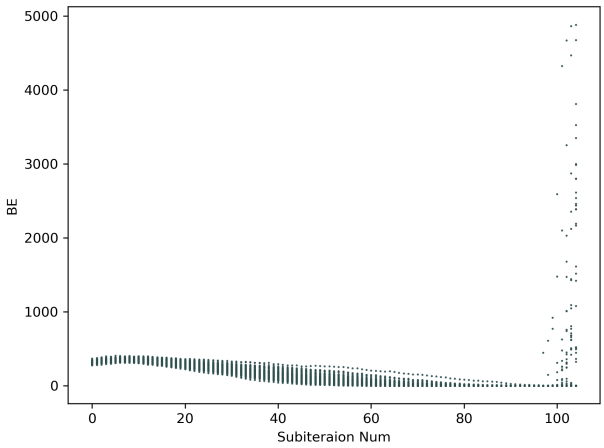


Figure 1. Count of error bits in decoding.

3. Design and Implementation

The hardware architecture of the GPON decoder based on the improved reordered algorithm is shown in Figure 2. This structure can be applied to the design and implementation of various quasi-cyclic LDPC codes, and it is a universal hardware design structure for LDPC decoders that can achieve efficient decoding for different regular or irregular quasi-cyclic LDPC codes with various parity-check matrices.

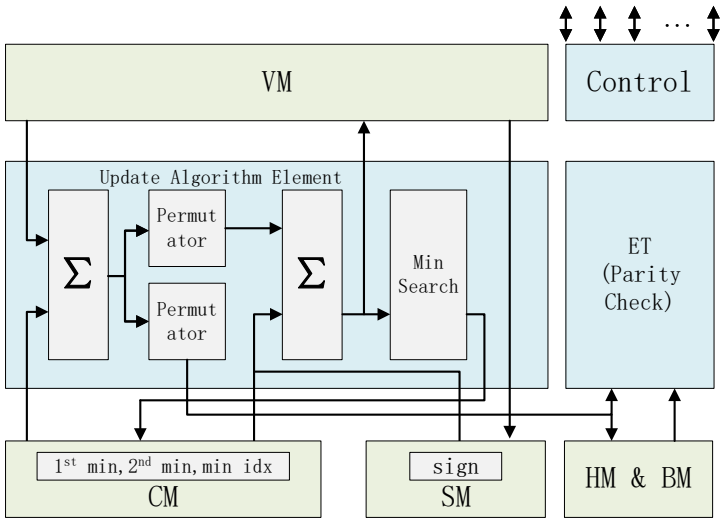


Figure 2. Decoder architecture.

A complete decoder core consists of a central controller (Controller), a variable node memory (VM), a check node memory (CM, SM), an update algorithm computation unit (Update Algorithm Element), a hard-decision code word memory (HM), and an early termination module (ET).

In the designed decoder, two memory modules (CM and SM) are used to store the complete check information. CM stores the minimum value (1st min), the second minimum

value (2nd min), the position information of the minimum value (min idx), and the sign product of all connected VN. SM stores the sign of each connected VN.

3.1. Partially Block-Parallel

A compromise between serial-layered decoding and parallel-layered decoding is considered—more blocks are processed per cycle, but multiple cycles are still required to process a sublayer. Since more blocks are processed per cycle, but the blocks of each layer will not be processed in one cycle and are not fully parallelized, it is called partially block-parallel (PBP). Compared with the serial design, the hardware implementation of partially block-parallel hierarchical decoding consumes more hardware resources than the serial decoder but takes more throughput compared with the parallel design, lowering complexity and making the implementation of the long-code decoder possible. In this design, the decoder is able to process 3 blocks at the same time.

In the decoder proposed in [15], the VM only stores amplitude information, and the sign of the VN information is obtained by reading the SM. In the reordered algorithm, the update of VN information requires 2 read and 1 write operation for SM; so, they use a memory bank design, dividing the base matrix into 3 blocks so that the decoder designed in their work can access the SM simultaneously without conflict. The path of VN sign during the operation of the decoder is shown in Figure 3a: (1) It is extracted from memory to recover VN information and CN information. (2) Then, sign is extracted again to recover CN information. (3) Finally, after computing the updated VN information, its sign part is stored in the SM.

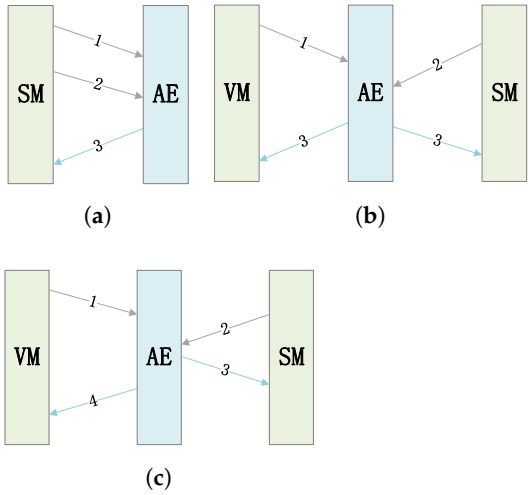


Figure 3. Path of VN sign. (**a**) When the variable memory only stores the copy information, the moving path of the variable node symbol information; (**b**) When variable memory stores symbol information, the moving path of variable node symbol information; (**c**) This paper presents a new method to eliminate duplicate storage.

In the PBP decoder, memory access operations increase by several times; in this design, they need to be increased by 3 times. Therefore, the sign memory needs to be accessed 9 times per cycle ($3 * 3 = 9$). However, dividing the base matrix into 9 blocks will introduce a large number of bubbles, which will significantly reduce the throughput. This paper tries to reduce the read/write access number to 2 for each block, then dual-port memory in combination with dividing the base matrix into 3 blocks allows for 6 read/write accesses per cycle.

First, assuming that sign is stored in both sign memory and VN memory, the movement of variable symbol information is shown in Figure 3b: (1) VN information is directly obtained from VM, and sign part is used to restore the CN information. (2) Sign extracted from SM is used to recover CN information. (3) After the update is completed, the sign

part of the updated VN information needs to be written into both VM and SM. Obviously, using this method, each block update only requires one read/write access. However, since the storage share in [15] is abandoned, the symbol part of VN information in VM is duplicated in sign memory; so, the decoder needs to increase memory bits by one code length (17,664 bits).

By analyzing Figure 3b, it can be observed that the sign in SM is only used to recover the second CN information R_{mn}. The sign part written into the VM is used to compose the VN information and restore the first CN information $R_{m'n}$ at the same time. Consider a method for re-eliminating duplicate storage as follows: After completing block updates, the sign part of the updated variable node VN is firstly stored in VM, and when it is read out from VM, it is written into SM. Finally, when the sign is needed, it is read out from SM, and the data flow can be summarized as shown in Figure 3c.

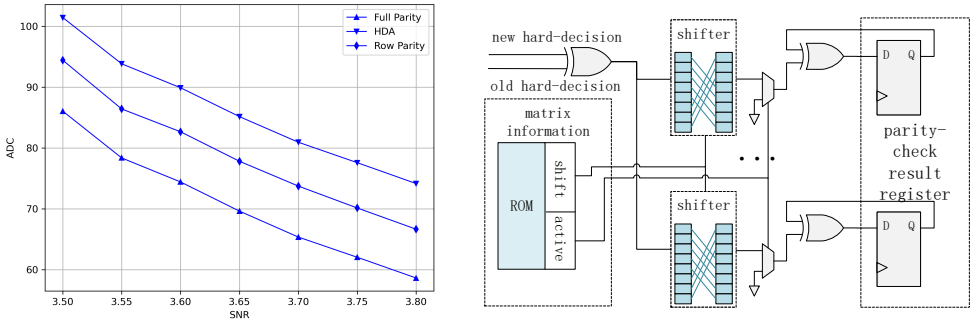
The sign read–write process is as follows: Assuming that the degree of a column is n, it is obvious that n signs need to be stored. The 1st is stored in VM, and the remaining $n-1$ signs are stored in SM. During the update, the 1st is read out from VM, the old 2nd is read out from SM, and then the 1st read out from VM is written into the position of the 2nd in SM. After the update is completed, the 2nd is obtained and written into VM. The later update is similar. It can be observed that for the same column, since SM only has space to store $n-1$ signs, the n signs are cyclically stored in $n-1$ position in SM; so, the read and write addresses of SM are not fixed and will change continuously during decoding.

3.2. Decoding Early Termination

To end the decoding as early as possible after successful decoding, researchers have proposed various mechanisms for early termination (ET) decoding iterations [16,17].

Among them, full parity check is the most obvious. Hard-decision aid (HDA) [17] takes advantage of the fact that the hard-decision result remains relatively stable when the decoding converges, with simple hardware implementation. Row parity check processes layer by layer, which may cause an error floor [18].

Figure 4a gives the comparison of the average decoding cycles (ADC) of the 3 methods. SNR is the multiple signal-to-noise ratio points tested, and ADC is the average number of decoder iterations corresponding to the early termination decoding method, which corresponds to the average decoding time in the circuit. As can be seen, the average time for full parity is the smallest, the row parity check is slightly larger, and HDA consumes the most time.



(**a**) (**b**)

Figure 4. (**a**) Average decoding cycles comparison. (**b**) Serial full parity-check circuit.

Due to the BER performance indicator for 50G-PON (less than 10^{-12}), in this paper, we decide to use full parity check. However, the traditional full parallel parity-check method requires the construction of a large parity tree structure, which consumes a lot of circuit area resources. In this paper, a serial full parity-check method is proposed to reduce the complexity by using the cyclic shift characteristic of the parity matrix.

As shown in Figure 4b, before the decoder begins decoding, all the parity-check result registers are reset to 0. After the decoding starts, the new hard-decision message and the

old hard-decision message pass through an XOR (exclusive OR) gate to obtain the changed value of the decision message. Then, according to the matrix information stored in ROM, the shifter is controlled to shift the updated decision message, and the updated decision message is XOR with the original value of the parity-check result registers to update the value of the parity-check result registers. The value of the parity-check result registers reflects the check result of the corresponding row, and its value is updated synchronously every cycle. When the values of the parity-check result registers are all 0, it means that the check is successful but considering the conditions have been met at the beginning of decoding; so, only when the number of iterations of the decoder is greater than 1, the judgment is made.

3.3. Back-Up Memory

In the previous section, this paper discussed the error caused by the quantized overflow processing of VN information in the decoder. After tracing the error bits in several decoding processes, a rapidly increasing number of error bits was observed in the late decoding period. If the hard decision is output in this case, it will output a large number of error bits, extremely hindering the performance of the decoder.

To this end, this paper adds an additional hard-decision memory—back-up memory (BM). Its mechanism is shown in Figure 5. In the early stage of decoding, both BM and HM are updated, and the content stored in BM is the copy of HM. In the later stage of decoding, when the decoder is detected to near convergence, it waits for a period of time. If the decoder does not finish decoding within this period, it is considered that a large number of overflow errors will occur in the decoder. The update of BM is disabled and only HM is updated. At the end of the decoding, output from HM is selected if the decoder successfully interprets, output from BM is selected if the decoding fails and an operation to disable BM is triggered, and output from HM is selected if the decoding fails and no operation to disable BM is triggered.

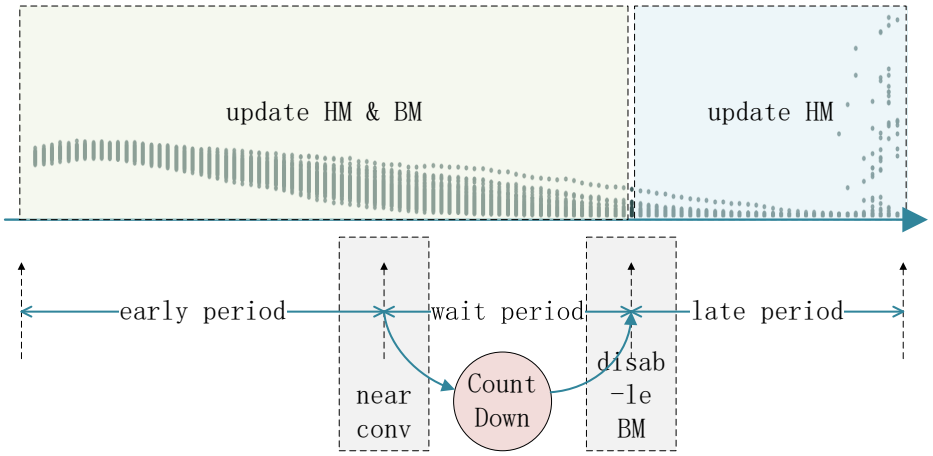


Figure 5. BM mechanism.

4. Results

4.1. Implementation Resource Utilization

The 50G-PON standard LDPC decoder designed in this paper is implemented using the VC707 development board of the Virtex-7 series, with the specific FPGA model number xc7vx485t. The resource utilization of the designed LDPC decoder based on the synthesis and implementation of this development platform is shown in Table 2.

Table 2. Decoder FPGA Implementation Comparison.

Design FPGA	[19] **Virtex-7 VC707**	[20] **Ultrascale-XCVU125**	**PBP Virtex-7 VC707**
code length	(1280 1536 2048)	17,664	17,664
code rate	1/2 2/3 4/5	57/69	57/69
algorithm	NMS	NMS	NMS
FF	13,274	675,854	76,012
LUT	32,560	537,504	144,086
iteration cycle num	73/109/181	N/A	96
sub-matrix num	60/96/156	275	275
max iteration num	6	12	12
f (MHz)	180	75	100
throughput (Gbps)	0.320	1.200	1.235

Table 2 gives the resource consumption of the designed decoder. Based on the 50G-PON standard, this design achieves a similar throughput to that of the paper in [20], with about 0.27 times the LUT and about 0.11 times the FF. Compared with the paper in [19], this design achieves about 3.86 times the throughput, with about 4.43 times the LUT and about 5.73 times the FF. On the one hand, this is due to the large code length of the 50G-PON, so the FPGA frequency is limited, and on the other hand, the throughput is lower because the maximum number of iterations is twice that in [19].

Table 3 gives the synthesis results of the implemented LDPC decoder. It can be seen that while also based on the TSMC 65 nm process, the designed decoder has more than twice the throughput area rate(TAR) of [21] and more than three times the TAR of [19], proving that the designed circuit structure can effectively improve the area efficiency of the optical communication decoder circuit.

Table 3. Synthesis Result Comparison.

Design Process	[19] **TSMC 65 nm**	[21] **TSMC 65 nm**	**PBP TSMC 65 nm**
code length	(1280 1536 2048)	8176	17,664
code rate	1/2 2/3 4/5	7/8	57/69
algorithm	NMS	NMS	NMS
area (mm^2)	0.819	2.3	2.610
f (MHz)	500	380	800
throughput (Gbps)	0.887	4.1	9.864
TAR ($Gbps/mm^2$)	1.083	1.783	3.779

Throughput Area Rate(TAR).

4.2. BER Performance Test

After punching the last 1.5 columns of the parity-check matrix, the decoding performance of the 6-bit quantized decoder is tested by FPGA, the BER/FER curve shown in Figure 6 is obtained, and Table 4 is the conversion of the input BER with the corresponding σ and SNR. From Figure 6, it can be seen that at an input BER of 2.3×10^{-2}, 11 iterations were able to achieve an output BER lower than 1×10^{-12} with a coding gain of about 10.4 dB, which is sufficient to meet the demand of optical communication for output error rate.

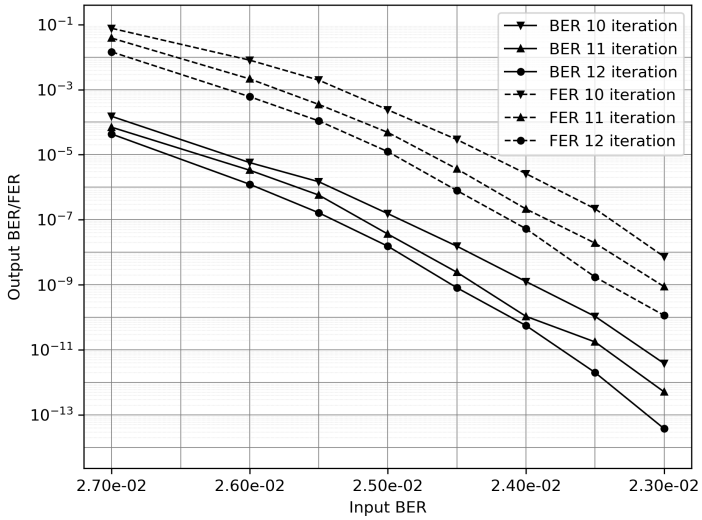


Figure 6. BER/FER performance.

Table 4. BER and corresponding σ and SNR.

BER	Noise (σ)	SNR (dB)
2.30×10^{-2}	5.0115×10^{-1}	3.7246
2.35×10^{-2}	5.0345×10^{-1}	3.6849
2.40×10^{-2}	5.0572×10^{-1}	3.6457
2.45×10^{-2}	5.0798×10^{-1}	3.6071
2.50×10^{-2}	5.1021×10^{-1}	3.5690
2.55×10^{-2}	5.1243×10^{-1}	3.5313
2.60×10^{-2}	5.1463×10^{-1}	3.4940
2.70×10^{-2}	5.1899×10^{-1}	3.4209

5. Conclusions

In this paper, we propose an area efficient LDPC decoder without loss of performance. In the hardware implementation process, this paper completed the overall circuit design of the decoder with the goal of improving area efficiency and ensuring decoding performance and built an FPGA-based decoding performance verification platform. We partially adopted block-parallel processing so that the decoder can update up to three blocks per cycle at the same time, which greatly reduces the iteration time and improves the throughput of the decoder. By adjusting the update order of matrix block, the storage space reuse is realized, and the utilization rate of hardware resources and area efficiency is improved. The performance loss of line-by-line check is solved by using the full parity check early-termination decoding method. Finally, we add an additional hard-decision memory to solve the decoding oscillation caused by overflow errors.

By parallel processing of three submatrices and storage reuse of node information, 9.864 Gbps was achieved based on TSMC65 nm synthesis with 800 MHz circuit frequency in and area of 2.61 mm^2, with about two to four times TAR improvement. By proposing a mechanism of spare decision storage to avoid errors caused by quantization overflow of the decoder at an input BER of 2.3×10^{-2} (SNR about 3.72 dB), the output BER was lower than 10^{-12}.

Author Contributions: Methodology and investigation, J.W.; methodology, writing—original draft, J.Y.; validation, G.Z.; software support, X.Z.; project administration and funding acquisition Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key R&D Program of China under No.2019YFE0120700, the National Natural Science Foundation of China under Grant 62274041.

Data Availability Statement: The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

PON	Passive Optical Network
LDPC	Low-Density Parity Check
FPGA	Field-Programmable Gate Array
TSMC	Taiwan Semiconductor Manufacturing Company
BER	Bit Error Ratio
SNR	Signal-to-Noise Ratio
MN	Mackay–Neal
SPA	Sum-Product Algorithm
TPMP	Two-Phase Message Passing
BP	Belief-Propagation
MSA	Min-Sum Algorithm
CN	Check Notes
VN	Variable Notes
TAR	Throughput Area Rate
FIFO	First in First Out
PBP	Paritially Block-Parallel
ET	Early Termination
HDA	Hard-Decision Aid
ADC	Average Decoding Cycles
OMS	Offset Min-Sum
NMS	Normalized Min-Sum
TDMP	Turbo-like Decoding Message Passing
FF	Flip Flop
LUT	Look-Up Table
FER	Frame Error Ratio

References

1. Gallager, R. Low-density parity-check codes. *IRE Trans. Inf. Theory* **1962**, *8*, 21–29.
2. MacKay, D.J.; Neal, R.M. Near Shannon limit performance of low density parity check codes. *Electron. Lett.* **1996**, *336*, 457–458. [CrossRef]
3. MacKay, D.J.C. Good error-correcting codes based on very sparse matrices. *Proc. IEEE Int. Symp. Inf. Theory* **1999**, *45*, 399–433. [CrossRef]
4. MacKay, D.J.C.; Wilson, S.T.; Davey, M.C. Comparison of constructions of irregular Gallager codes. *IEEE Trans. Commun.* **1999**, *47*, 1449–1454. [CrossRef]
5. Eleftheriou, E.; Mittelholzer, T.; Dholakia, A. Reduced-complexity decoding algorithm for low-density parity-check codes. *Electron. Lett.* **2001**, *37*, 102–104. [CrossRef]
6. Fossorier, M.P.; Mihaljevic, M.; Imai, H. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Trans. Commun.* **1999**, *47*, 673–680. [CrossRef]
7. Chen, J.; Dholakia, A.; Eleftheriou, E.; Fossorier, M.P.; Hu, X.Y. Reduced-complexity decoding of LDPC codes. *IEEE Trans. Commun.* **2005**, *53*, 1288–1299. [CrossRef]
8. Chen, J.; Fossorier, M.P. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Trans. Commun.* **2002**, *50*, 406–414. [CrossRef]
9. Mansour, M.M.; Shanbhag, N.R. High-throughput LDPC decoders. *IEEE Trans. Very Large Scale Integr. Syst.* **2003**, *11*, 976–996. [CrossRef]

10. Gunnam, K.K.; Choi, G.S.; Yeary, M.B.; Atiquzzaman, M. VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 4542–4547.
11. Gentile, G.; Rovini, M.; Fanucci, L. Low-Complexity Architectures of a Decoder for IEEE 802.16e LDPC Codes. In Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), Lubeck, Germany, 29–31 August 2007; pp. 369–375.
12. Zhang, J.; Fossorier, M.P. Shuffled iterative decoding. *IEEE Trans. Commun.* **2005**, *53*, 209–213. [CrossRef]
13. Sharon, E.; Litsyn, S.; Goldberger, J. Efficient Serial Message-Passing Schedules for LDPC Decoding. *IEEE Trans. Inf. Theory* **2007**, *53*, 4076–4091. [CrossRef]
14. Boncalo, O.; Kolumban-Antal, G.; Amaricai, A.; Savin, V.; Declercq, D. Layered LDPC Decoders with Efficient Memory Access Scheduling and Mapping and Built-In Support for Pipeline Hazards Mitigation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 1643–1656. [CrossRef]
15. Lee, H.C.; Li, M.R.; Hu, J.K.; Chou, P.C.; Ueng, Y.L. Optimization Techniques for the Efficient Implementation of High-Rate Layered QC-LDPC Decoders. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 457–470. [CrossRef]
16. Wu, Y.S.; Lin, C.H.; Lin, S.Y. Ultra-low complexity early termination scheme for layered LDPC decoding. In Proceedings of the 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE), Tokyo, Japan, 7–10 October 2014; pp. 711–712.
17. Li, J.; He, G.; Hou, H.; Zhang, Z.; Ma, J. Memory efficient layered decoder design with early termination for LDPC codes. In Proceedings of the 2011 IEEE International Symposium of Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011; pp. 2697–2700.
18. Blad, A.; Gustafsson, O.; Wanhammar, L. An early decision decoding algorithm for LDPC codes using dynamic thresholds. In Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005, Cork, Ireland, 2 September 2005; Volume 3, pp. 285–288.
19. Xie, J. LDPC Encoder and Decoder for CCSDS Deep-Space Standard. Master's Thesis, Fudan University, Shanghai, China, 2020.
20. Wang, W.; Tao, K.; Qian, W.; Cai, Y.; Lei, M.; Zhou, X.; Chien, H.C.; Liang, J.; Zhang, S.; Liu, Z. Real-Time FPGA Verification for 25G-PON and 50G-PON LDPC Codes. In Proceedings of the Conference on Lasers and Electro-Optics (CLEO), San Jose, CA, USA, 10–15 May 2020; pp. 1–2.
21. Li, S.; Zhang, Q.; Chen, Y.; Zeng, X. A High-Throughput QC-LDPC Decoder for Near-Earth Application. In Proceedings of the 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), Shanghai, China, 19–21 November 2018.

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.