Journal of Electrotechnology, Electrical Engineering and Management (2017) Vol. 1, Number 1
Clausius Scientific Press, Canada

FPGA Implementation of LDPC Decoder with Low Complexity

Li Lintao[1, a], **Zheng Hao**[2,b]

[1] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing; 100876, China

[2]Schoold of Information and Electronics, Beijing Institute of Technology, Beijing; 100081, China

[a]llintao@bupt.edu.cn, [b]3120130330@bit.edu.cn

Keywords: LDPC; Decoder; Low Complexity

Abstract. According to the limitation of resources on satellite, this paper focuses on the design and realization of low complexity LDPC decoder. A new implementation method of LDPC decoder is proposed, a various kinds of LDPC codes could be supported. Finally, a (4096, 2048) LDPC decoder is implemented for verification based on a Xilinx Vertex4 xc4vsx35 FPGA platform. The implementation result shows that only 4% FPGA logic resources were consumed and the maximum clock frequency could achieve 180MHz.

1. Introduction

The low-density parity-check (LDPC) code was proposed by Gallager[1] in 1962. Such codes are linear block codes with low density parity check matrix. Since its rediscovery in the 1990s, the LDPC code has been widely applied to DVB-S2 and wireless metropolitan area networks (IEEE 802.6) because of its excellent error correction performance and easy-to-implement decoding structure. A corresponding LDPC coding scheme is proposed by the Consultative Committee for Space Data System for near-earth and deep-space application environments. Existing research on LDPC codes focuses mainly on the design and engineering realization of LDPC encoder and decoder. LDPC decoder design with low-complexity realization is the basis and prerequisite of LDPC code for satellite applications customized for the limited payload resources of the satellite.

The three main realization structures of LDPC decoder are serial, full parallel, and partial parallel structures. The serial decoding structure presents low realization complexity but long decoding delay and low throughput. The full parallel structure exhibits low decoding delay, large throughput, and high complexity; however, it is difficult to achieve the hardware implementation. The partial parallel structure can achieve a good balance between realization complexity and decoding speed. Thus, this decoding structure is commonly applied. In 2006, Verdier[2] proposed a generalized decoding structure based on the modified minimum sum (Min-sum) algorithm. A parallel decoding algorithm of specific pseudo-random rule LDPC codes and irregular LDPC codes is implemented on the ALTERA APEX 20 Ke FPGA platform. The decoder consumes 1591 logic cells and 160 KB of space, and the decoder can work at the maximum clock rate of 37.89 MHz. Arnone et al.[3] analyzed and compared the implementation complexity and performance difference of decoder based on logarithmic sum-product and the simplified soft Euclidean distance iterative decoder. A low-complexity FPGA realization structure is thus proposed for the two decoding algorithms. In addition, some LDPC decoders are proposed in References [4–8].

In this study, a low-complexity LDPC decoding method is proposed. The decoder uses the normalized Min-sum algorithm (NMSA). By using a dual-port Block RAM of FPGA chip, a general low-complexity two-channel LDPC decoder is designed. The decoder can support LDPC with multiple bit rates and codes. In the end, the design based on Xilinx xc4vsx35 chip is experimentally verified, and the results are compared with the traditional serial decoders and partial parallel decoders. The results show that the hardware resource consumption of the LDPC decoder designed by the proposed method is obviously less than that of traditional serial and partial parallel decoders. Therefore, the proposed decoder structure has certain engineering practical value.

2. NMSA

NMSA introduced by Chen and Fossorier[9] is a simplified version of Log-BP algorithm, and the details are as follows.

1) Initialization

For AWGN channels, the variable node information can be initialized to

$$L(q_{ji}) = L(p_i^0) = y_i\,,\ i = 1,2,3,\cdots,n \tag{1}$$

where y_i is the soft received value.

2) Check node update(CNU)

The CNU at the lth iteration can be expressed as follows:

$$L(r_{ji}^l) \approx \eta \cdot \Big(\prod_{i'\in N(j)\backslash i} \alpha(q_{ji'}^{l-1})\Big) \min_{i'\in N(j)\backslash i} \beta(q_{ji'}^{l-1}) \tag{2}$$

where η is a correction factor in the range of (0, 1), $\alpha(q_{ji'}^{l-1}) = sign(q_{ji'}^{l-1})$, and $\beta(q_{ji'}^{l-1}) = \left|q_{ji'}^{l-1}\right|$.

3) Variable node update(VNU)

The VNU at the lth iteration can be expressed as follows:

$$L(p_i^l) = L(p_i^0) + \sum_{j\in M(i)} L(r_{ji}^{l-1}) \tag{3}$$

$$L(q_{ji}^l) = L(p_i^l) - L(r_{ij}^{l-1})\,. \tag{4}$$

4) Decision

After completing each iteration of the VNU, a decision is made on $L(p_i^l)$. If $L(p_i^k) \geq 0$, then $\hat{x}_i = 0$; otherwise $\hat{x}_i = 1$. If $\mathbf{Hx}^T = 0$ holds or the maximum iteration number is reached, the decoding process ends; otherwise, the decoding continues from Step 2.

3. FPGA-based Decoder Design

An LDPC decoder is designed for reducing the realization complexity of the decoder and decreasing the hardware resource consumption of FPGA. In the design process, the decoder parallelism and the number of quantization bits are reduced. Also, the decoder structure is optimized. In the present study, a two-channel parallel decoding method is applied to completely utilize the dual-port Block RAM of the FPGA chip, and an LDPC decoder with low implementation complexity and low resource consumption is designed. The 6-bit quantization is used achieve a balance between storage requirements and performance.

A. Low-complexity Structure of LDPC Decoder

The proposed realization structure of LDPC decoder is shown in Fig.1.

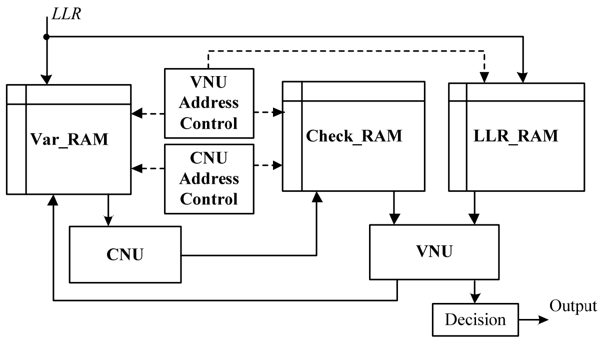

Figure 1. Realization structure of LDPC decoder

The LDPC decoder mainly consists of four parts: CNU module, VNU module, intermediate variable storage module, and memory read/write address control module. The intermediate variable storage module includes an initial log-likelihood ratio RAM (LLR_RAM), a check update result memory (Check_RAM), and a variable update result memory (Var_RAM); each of these memories is implemented using a dual-port Block RAM. The memory read/write address control module includes a CNU address control (Check_addr) and a VNU address control (Var_addr).

In the initialization process, LLR value is written to LLR_RAM and Var_RAM. The write address generated by the Var_addr module. When the check node is updating, the data are read from Var_RAM according to the address generated by the Check_addr; these data are then sent to the CNU module. The outputs of the CNU are written to the Check_RAM, and the write address is controlled by the Check_addr. When the variable node is updating, it first reads the data from the LLR_RAM and the Check_RAM according to the address generated by the Var_addr; these data are then sent to the VNU module. After the variable node is updated, a decision can be made. If the maximum number of iterations is reached, then the output of the decision will be the decoding result; otherwise, the update result is written to the Var_RAM and the next iteration is performed. Figure 2 shows the overall operation timing sequence of the LDPC decoder, and the read/write state of the three memories in the decoding process. The read/write of the two memories can use the same address generation module, thus simplifying the logic and reducing hardware resource consumption.

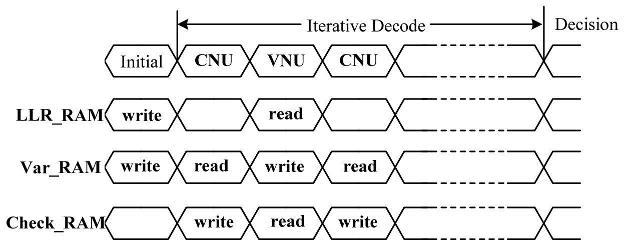

Figure 2. Operation timing sequence of the decoder

The decoding delay can be expressed as follows:

$$T = T_{initial} + (T_{CNU} + T_{VNU}) \times iter\_time \, , \tag{5}$$

where $T_{initial}$ is the time allotted for initialization; T_{CNU} and T_{VNU} are the time required to update the check node and the variable node in one iteration, respectively; *iter_time* is the iteration number.

B. CNU Module

This module includes absolute value comparison, multiplicative correction, and some other operations. The realization structure of CNU is shown in Figure 3. The CNU adopts a two-channel parallel way. First, the two inputs *din1* and *din2* of the module are separated by the modulus value, and the corresponding sign bit *sign1*, *sign2* and absolute value *beta1*, *beta2* are obtained. After caching the sign bit, the update can be conducted by a simple XOR operation using Equ. (2).

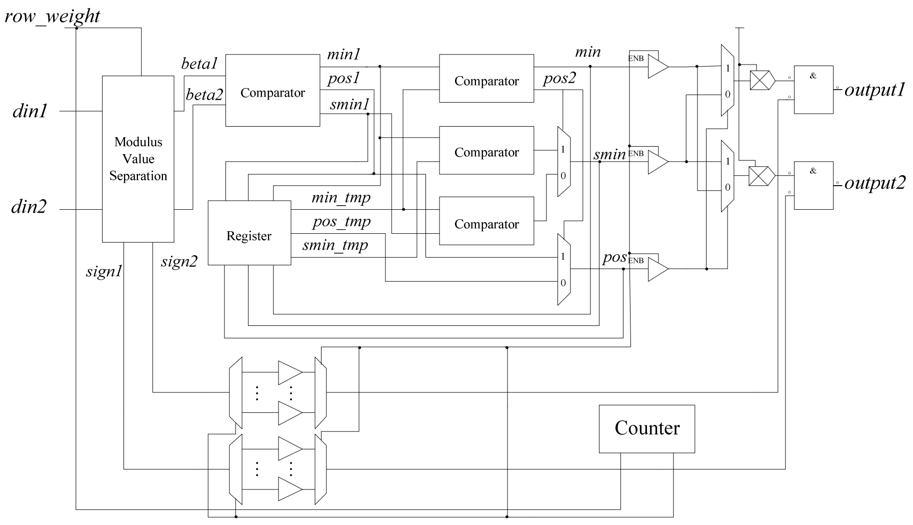

Figure 3. Check node update module structure

During the absolute value comparison operation, *beta1* and *beta2* are first compared after the modulus value separation; the minimum absolute value *min_temp*, minimum position *pos_tmp*, and second minimum value *smin_temp* are also cached. Thereafter, for each pair of absolute value input, a minimum value and a second minimum value are selected from *min1*, *smin1*, *min_temp* and *smin_temp* for those four values because the two sets of data satisfy

$$\begin{gathered} min1 \le smin1 \\ min\_temp \le smin\_temp \end{gathered}.$$

Therefore, the minimum value must be *min1* or *min_temp*. The operation of the second minimum value only requires the comparison of *min1* and *smin_temp*, as well as *smin1* and *min_temp*. If the minimum value is *min1*, then the second minimum must be *smin1* or *min-temp*. If the minimum value is *min_temp*, then the second minimum must be *min1* or *smin_temp*. In this way, the comparison operation of the minimum and second minimum values can be completed using four two-input comparators. As the correction coefficient is always set to 0.8, the multiplicative operation can be finished using bitwise shift operation and subtraction approximatively.

For a (n,k) LDPC code with a maximum row weight of a, the time required to complete the CNU is $\lceil a/2 \rceil \times (n-k)$ operating clocks; this time is less than half the time of the serial operation.

C. VNU Module

The initial LLR value and outputs of the CNU module are inputs for VNU module. Fig. 4 shows the implementation structure of VNU module. *dinc* represents the initial LLR value (i.e., $L(p_i^0)$), *dina* and *dinb* represent the check update result (i.e., $L(r_{ji})$). During the updating process, *dinc*, *dina*, and *dinb* are added on the basis of the column weight and *bit_sum* is the summation. Adding *bit_sum* with -*dina* and -*dinb*, the $L(p_i^k) - L(r_{ij}^{k-1})$ operation of the variable update can be completed; consequently, the variable update result can be obtained. *bit_dout* is the sign bit of *bit_sum* and is also the result of the decision output after completing an iterative decoding.

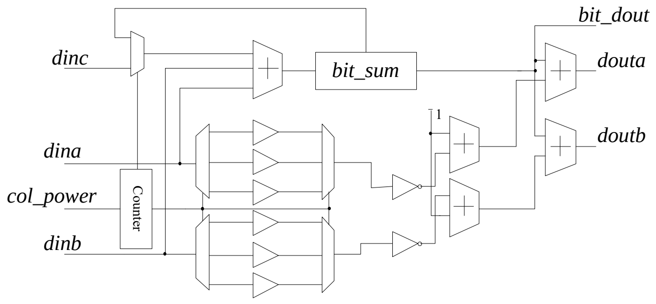

Figure 4. Variable node update structure

For an LDPC code with a maximum col weight of b, the time required to complete the variable node update is $\lceil b/2 \rceil \times n$ operating clocks; the time is less than half the time of that in a serial decoder.

D. FPGA Realization Result of LDPC Decoder

In this study, we use the irregular quasi-cyclic (4096, 2048) LDPC code as an example[10–11]. The maximum number of iterations of the decoder is set to 25.

Table 1. Resource evaluation for FPGA decoding realization

	16-channel parallel decoding structure	4-channel parallel decoding structure	Traditional Serial decoding structure	Decoding structure of this paper
LUTs	11,022 (36%)	7,243 (24%)	4,681 (15.5%)	849 (3%)
Slices	6,148 (40%)	3,890 (25%)	2,362 (15%)	555 (4%)
Block RAMs	36 (19%)	23 (12%)	23 (12%)	28 (15%)
Max frequency	121.7 MHz	120.3 MHz	140.5 MHz	181.2 MHz
Decoding delay	1.10 ms	1.76 ms	3.69 ms	2.30 ms

Table 1 shows the resource consumption of the FPGA implementation with different decoder structures under the given conditions. The proposed decoding implementation structure uses only 555 Slice logic resources and 28 Block RAM memories to achieve the realization of a (4096, 2048) LDPC decoder. The hardware resource consumption of the proposed decoder structure is significantly less than that of traditional partial parallel and serial decoding structures.

4. Conclusion

A hardware implementation structure of decoder with low complexity is proposed to reduce the hardware resource consumption of LDPC decoder and address the limitation of the payload resource in satellite communication system. Accordingly, a (4096, 2048) LDPC decoder is designed based on Xilinx Vertex-4 xc4vsx35 chip. The hardware resource consumption of this decoder is much less than that of traditional parallel and serial decoding structures. The FPGA implementation results show that the proposed decoder structure can effectively reduce the hardware resource consumption and has certain engineering application value.

Acknowledgements

This work was financially supported by the Natural Science Foundation (61327806, 61501028).

References

[1] Gallager R G. Low-density parity-check codes[J]. *IRE Transactions on Information Theory*, 1962, 8(1): 21-28.
[2] Verdier F, Declercq D. A Low-Cost Parallel Scalable FPGA Architecture for Regular and Irregular LDPC Decoding[J]. *IEEE Transactions on Communications*, 2006, 54(7): 1215-1223.
[3] Arnone L J, Castineira Moreira J, Farrell P G. Field programmable gate arrays implementations of low complexity soft-input soft-output low-density parity-check decoders[J]. *Communications*, 2012, 6(12): 1670-1675.
[4] Wang B, Wan L. FPGA Implement for High Performance and Low Complex LDPC Decoder[J]. Modern Electronics Technique, 2008(18): 135-138. (in Chinese)
[5] Zhu J, Zhang H B, Pan Y. A Low-complexity Iterative Decoding Algorithm for LDPC Codes[J]. Telecommunication Engineering, 2006(5): 95-97. (in Chinese)
[6] Yin S S, Wang Z X. Low complexity encoding and decoding research of DVB-S2 LDPC[J]. Journal of Chongqing University of Posts & Telecommunications, 2012, 24(4): 457-461.
[7] Xiaoheng Chen, Qin Huang, Shu Lin, Akella V. FPGA-Based Low-Complexity High-Throughput Tri-Mode Decoder for Quasi-Cyclic LDPC Codes [C]. Communication, Control and Computing, Monticello: IEEE, 2009: 600-606.
[8] Daesun Oh, Parhi K K. Low Complexity Implementations of Sum-Product Algorithm for Decoding Low-Density Parity-Check Codes [C]. Signal Processing Systems Design and Implementation, Banff, Alta: IEEE, 2006: 262-267.
[9] Chen J H, Fossorier M P C. Density evolution for two improved BP-based decoding algorithms of LDPC codes[J]. IEEE Communications Letters, 2002, 6(5): 208-210.
[10] Y. Kou, S. Lin, and M.P.C. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. IEEE Trans. Inform. Theory, 47(7):2711–2736, Nov. 2001.
[11] J. Kang, Construction, Decoding and Application of Low-density Parity-check Codes. University of California, Davis, 2009