Dynamic Maximum Iteration Number Scheduling LDPC Decoder for Space-Based Internet of Things

Ruijia Yuan[1(✉)], Tianjiao Xie[1,2], and Yi Jin[1]

[1] China Academy of Space Technology (Xi'an), Xi'an, People's Republic of China
yuanyuanruijia@163.com
[2] School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China

Abstract. For Space-based internet of things (S-IoT) application scenario, a Dynamic Maximum Iteration Number (DMI) scheduling decoder for LDPC codes is proposed. Distinct from traditional Static Maximum Iteration Number (SMI) scheduling LDPC decoder using fixed maximum iteration number, our DMI LDPC decoder has extra circuit to obtain the dynamic maximum iteration number, which could improves BER performance only at the expense of a slightly logic resources and a small ratio of memories, compared with conventional SMI scheme. Therefore, the DMI decoder is very suitable for the fluctuation of signal-to-noise ratio of S-IoT link.

Keywords: Space-based Internet of Things (S-IoT) · LDPC decoder · Iteration · FPGA

1 Introduction

The Internet of Things (IoT) [1] is a network application mode for machine-to-machine communication and interconnection. The space-based Internet of Things (S-IoT) [2] refers to the coverage of blind areas in remote areas, oceans and other terrestrial networks, and relies on satellite communication networks to achieve reliable communication transmission. The terminals of the S-IoT are widely distributed in land, sea, air, and sky. The propagation delays of various types of terminals to satellites are not only long but also jagged, resulting in different signal losses. Therefore, the specificity of the link is that the signal-to-noise ratio (SNR) fluctuates greatly. Aiming at this channel characteristic of S-IoT, this paper proposes a variable maximum iterations design for LDPC decoder. This method can implement dynamic data for each frame of data in the decoding process according to the actual channel characteristics. The maximum iterations is allocated for different SNR to improve the BER performance of the system.

B. Li et al. (Eds.): IoTaaS 2019, LNICST 316, pp. 235–241, 2020.
https://doi.org/10.1007/978-3-030-44751-9_20

In current decoders [4–8], the iterations of the next block do not start until a preset maximum iteration number is reached, which is called Static Maximum Iteration Number (SMI) scheduling decoder in this paper. However, in SMI, low BER frames of decoder input can be corrected after a small number of iterations, thus the process circuit is idle. Conversely, that of high BER frames cannot be corrected until SMI is reached, thus it is potentially harmful to the overall BER performance of decoder. If the idle time saved by low BER frames can be effectively used by those high BER frames, the BER performance should be improved. Traditionally, stopping criteria [4–8] is only used to save energy in iterative decoders. However, authors in [3] apply this method only for Turbo soft-input/soft-output decoders and additional three data buffers at the decoder input is analyzed to increase the average number of iterations.

In this paper, we consider making full use of processing capability of LDPC decoders. Thus, additional more data buffers are also involved to further increase the average number of iterations. Moreover, the implementation circuit of generating dynamic maximum iteration number is also presented in this paper. Our DMI scheduling LDPC decoder could improve the BER performance while it maintains nearly the same hardware resources as the existing SMI scheme.

2 Static Maximum Iteration Number (SMI) Scheduling Decoder for LDPC Codes

2.1 LDPC Decoding Algorithm

The typical LDPC decoding algorithm is Sum-Product Algorithm (SPA) [4], which provides a powerful method for decoding LDPC codes. However, it suffers from large complexity and is very sensitive to finite word length implementation [6]. The Min-Sum decoding Algorithm (MSA) [5] is similar to the SPA, just with an approximation of check node process. The most popular derived MSA used in hardware implementation is normalized MSA (NMSA) [5], in which at the i-th iteration the Check Node processing Units (CNU) compute the Check to Variable (C2V) messages $R_{mn}^{(i)}$ as follows:

$$R_{mn}^{(i)} = \alpha \prod_{n' \in N(m) \backslash n} sign\left(Q_{n'm}^{(i-1)}\right) \min_{n' \in N(m) \backslash n} |Q_{n'm}^{(i-1)}| \tag{1}$$

Where normalized factor α is introduced to compensate for the performance loss in the MSA compared to SPA. Q_{nm} is the Variable to Check (V2C) messages, $N(m)$ denotes the set of variable nodes connected to check node m. The exclusion of an element n from $N(m)$ is denoted by $N(m) \backslash n$.

At the i-th iteration, the Variable Node processing Units (VNU) compute $Q_{nm}^{(i)}$ and $Q_n^{(i)}$ as the following:

$$Q_{nm}^{(i)} = C_n + \sum_{m' \in M(n) \backslash m} R_{m'n}^{(i)} \tag{2}$$

$$Q_n^{(i)} = C_n + \sum_{m' \in M(n)} R_{m'n}^{(i)} \tag{3}$$

Where Q_n represent the posterior message corresponding of variable node n. $M(n)$ denotes the set of check nodes connected to variable node n. The exclusion of an element m from $M(n)$ is denoted by $M(n)\backslash m$. C_n denotes the intrinsic (channel) message associated with variable node n.

$$C_n = log(P(x_n = 0|y_n)/P(x_n = 1|y_n)) \tag{4}$$

Here both C2V and V2C messages are called extrinsic messages. Decoding can repeat iteratively until the check equations are satisfied or a preset maximum iteration number is reached.

2.2 Static Maximum Iteration Number (SMI) Scheduling LDPC Decoder

Before we present a new DMI scheduling LDPC decoder, timing and its corresponding definitions are first analyzed.

From the above description of LDPC decoding algorithm, it can be seen that the calculations for each iteration need the channel messages of the intact one frame. So the incoming data should be stored into a buffer until one frame data is completely reached. Then the iterative decoding process can be executed. In engineering, for real-time decoding, conventional iterative decoders usually employ two independent buffers at the input of decoder. The capacity of each buffer is set to one frame size. One buffer is filled with current frame coming from the demodulator and the other is used for the iterative decoding process for last frame. Afterwards the buffers tasks are switched. As a result this method effectively avoids the previous frame data being covered with the current frame data before it is completely processed.

Therefore maximum processing time T_{mp} of each frame should be no more than the incoming time of one frame. Here T_{mp} can be calculated as

$$T_{mp} = I_m \times C_I \tag{5}$$

where I_m is maximum iteration number and C_I is the clock cycles per iteration.

Figure 1 illustrates the timing of existing SMI method for consecutive data of LDPC iterative decoder. It can be seen that T_{mp} (or I_m multiples C_I) of each frame is a constant, which approximates to the incoming time of one frame. So it is a Static Maximum Iteration number (SMI) strategy decoding, where both we_ramf1 and we_ramf2 are write-enable signals.

3 Proposed Dynamic Maximum Iteration Number (DMI) Scheduling LDPC Decoder

In the SMI method, I_m is a constant which is proportional to the time of storing one dataframe (Eq. (5)). Decoding can repeat iteratively I_m times until the entire

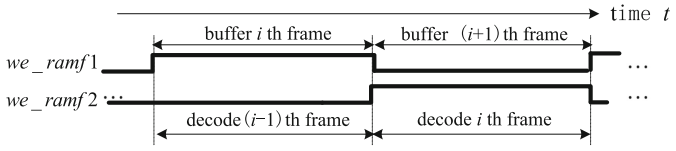

Fig. 1. The timing of SMI strategy decoding for consecutive data.

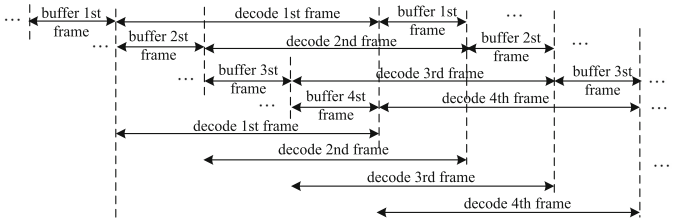

Fig. 2. The timing of DMI strategy decoding for consecutive data with F = 4 frames buffer.

codeword $\hat{x}$ (one frame) satisfy the parity check equations or the preset I_m is reached. However, some frames terminate the decoding after a small number of iterations, while other frame cannot be corrected either when I_m is reached. If the iterations frames saved idle time can be effectively used by those big iterations frames, the BER performance should be improved. By the observation, DMI strategy is proposed in this section.

In order to analyse how to use the saved idle time in DMI strategy, the timing of the DMI decoding with F = 4 frames buffers is demonstrated as an example in Fig. 2. Here the number of frames buffer F is an integer greater than 2. Because four buffers are utilized, each data frame can be retained in buffer for other three incoming frames time until a new data frame arrived. Therefore, the maximum iteration number prolongs three times more than the existing of buffering two frames case, i.e., maximum iteration number is increased up to $(\mathrm{F}-1)\times I_m$. Moreover, in the worst case, each frame can iterate minimum one frame incoming time, i.e., maximum iteration number is not less than I_m, which guarantees the BER performance of DMI not worse than that of SMI. This iterative time allocation method makes full use of idle time of that early termination frames to optimize the utilization of circuit resources. As a result maximum iteration number of DMI is dynamic, denoted I_{md}, which has a range from I_m to $(\mathrm{F}-1)\times I_m$, as shown in Fig. 2.

The main difference between SMI and DMI strategies are that the former has a predetermined constant I_m before the decoder is started, while for the latter its I_{md} of current frame should be calculated by the last previous frame. In fact, SMI specifies F = 2, which is a special case of DMI.

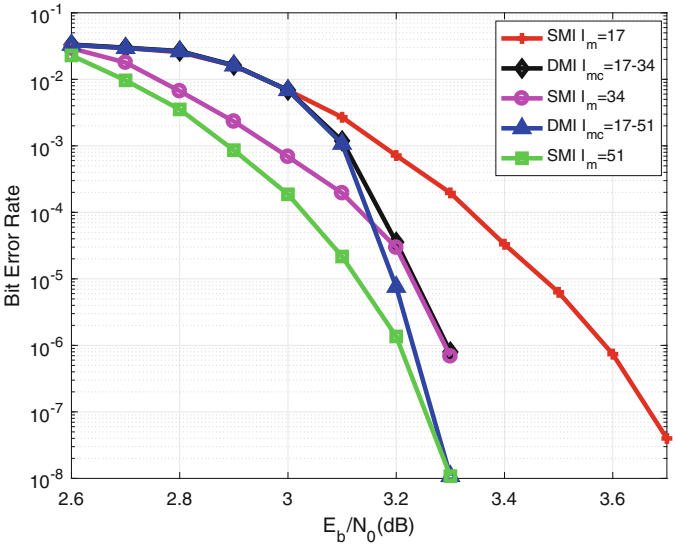

Fig. 3. BER curves of SMI and DMI for CCSDS 4/5 LDPC code

4 Simulation Results

In order to assess the performance of the above two scheduling decoding algorithms, SMI and DMI, LDPC codes in the CCSDS standard [9] was employed. AWGN channel is considered. In order to obtain a tradeoff between complexity and BER performance, the NMSA is used for check-node update. Normalization factor α is set to be 0.75.

The BER performance versus E_b/N_0 for the existing SMI and the new DMI are demonstrated in Fig. 3. It depicts the performance of SMI in solid lines with $I_m = 17$, 34 and 51, respectively. In order to make some comparisons with SMI, the DMI scheme with F = 3 and F = 4 frames buffer, corresponding $I_{md} =$ 17–34 and 17–51, respectively, are also illustrated in dashed lines in the same figure.

In Fig. 3, it can be seen that, At BER of 10^{-6}, our DMI with $I_{mc} = 17$–34 achieves the approximate BER performance as that of SMI with $I_m = 34$. The similar observations can be made that DMI with $I_{mc} = 17$–51 achieves the approximate BER performance as that of SMI with $I_m = 51$.

5 Implement Results

Based on the proposed architectures described in previous sections, we implement the CCSDS LDPC decoder. Q = 6 bits quantization scheme is adopted in our implementation. We choose the Xilinx Vertex5 xc5vlx330-1ff1760 device as the target FPGA.

Table 1. Resources utilization statistics of SMI and different DMI decoders

Iterations	SMI 17	SMI 34	DMI 17–34	DMI 17–51
Slices	12182	24364	12497	12497
Buffers	2 frames	4 frames	3 frames	4 frames
Throughput	160 Mbps	160 Mbps	160 Mbps	160 Mbps

Through the ISE10.1 place and route simulation, plus 5ns constraint, the decoder designed with the four maximum iterations shown in Table 1 can satisfy the constraint. The clock frequency of the decoder can be set to 200 MHz. The memory buffers the input frame data and the output decoded result data to ensure that the data of the input and output decoders are continuous. Assume that the input decoder clock is clkin, and the output decoder clock is clk (ie, the clock at which the decoder operates), so the relationship between the throughput of the decoder and the system clock is $clk \times CodeRate$, for 4/5 rate LDPC (5120, 4096), as shown in Table 1, fixed maximum 34 iterations is about twice as large as FPGA logic resources fixed by a maximum of 17 iterations, but encoded at a bit error rate of 10^{-6} The gain can be increased by 0.3 dB. It can be seen that for a fixed maximum number of iterations, FPGA resources can be used in exchange for high coding gain; iterative 17–34 times, iterations 17–51 times and fixed maximum 17 iterations account for the same FPGA logic resources. The throughput is the same, but the iteration is 17–34 times faster than the fixed maximum 17 iterations. When the bit error rate is 10^{-6}, the coding gain can be increased by 0.3 dB, which is equivalent to the fixed maximum 34 iterations, and the iteration is 17–51 times. The fixed maximum 17 iterations can increase the coding gain by 0.35 dB at a bit error rate of 10^{-6}, achieving a coding gain comparable to a fixed maximum of 51 iterations.

It can be seen that this section can save about half of the slice resources with the same coding gain and the same throughput than the fixed maximum iteration number of decoding schemes. In addition, for the buffer of input and output, BRAM is adopted in FPGA. In order to improve the utilization of BRAM resources, we use dual-port BRAM, which can buffer one frame with A and B ports of dual-port BRAM respectively. Therefore, buffering 3 frames and The number of BRAMs used to buffer 4 frames is the same, so iterative 17–51 times is more advantageous than FPGA iterations 17–34 times.

6 Conclusion

In this paper, to improve the code gain of LDPC decoder, we propose a DMI scheduling LDPC decoder architecture, which achieves better BER performance than the existing SMI decoder. Base on the proposed architecture, the FPGA implementation result shows the hardware resources of CCSDS decoder is a slightly more than that of SMI case. The proposed DMI scheduling LDPC

decoding strategy is a promising candidate for space-based IoT link with big fluctuation of signal-to-noise ratio.

Acknowledgment. This research was supported by National Natural Science Foundation of China under Grant 61801377.

References

1. Feltrin, L., et al.: Narrowband IoT: a survey on downlink and uplink perspectives. IEEE Wireless Commun. **26**(1), 78–86 (2019)
2. Qian, Y., Ma, L., Liang, X.: Symmetry chirp spread spectrum modulation used in LEO satellite internet of things. IEEE Commun. Lett. **22**(11), 2230–2233 (2018)
3. Vogt, J., Finger, A.: Increasing throughput of iterative decoders. Electron. Lett. **37**(12), 770–771 (2001)
4. MacKay, D.J.C., Neal, R.M.: Near Shannon limit performance of low density parity check codes. Electron. Lett. **33**(6), 457–458 (1997)
5. Chen, J., Dholakia, A., Eleftheriou, E., Fossorier, M.P.C., Hu, X.Y.: Reduced-complexity decoding of LDPC codes. IEEE Trans. Commun. **53**, 1288–1299 (2005)
6. Wang, Z., Cui, Z., Jin, S.: VLSI design for low-density parity-check code decoding. IEEE Mag. Circuits Syst. **11**, 52–69 (2011)
7. Nguyen-Ly, T., Savin, V., Le, K., Declercq, D., Ghaffari, F., Boncalo, O.: Analysis and design of cost-effective, high-throughput LDPC decoders. IEEE Trans. VLSI Syst. **26**(3), 508–521 (2018)
8. Lu, Q., Fan, J., Sham, C.W., Tam, W.M., Lau, F.C.M.: A 3.0 Gb/s throughput hardware-efficient decoder for cyclically-coupled QC-LDPC codes. IEEE Trans. Circuits Syst. I, Reg. Papers **63**(1), 134–145 (2016)
9. CCSDS 131.1-O-2: Low density parity check codes for use in near-earth and deep space applications, September 2007