Complexity-Reduced Algorithms for LDPC Decoder for DVB-S2 Systems

Eun A Choi, Ji-Won Jung, Nae-Soo Kim, and Deock-Gil Oh

ABSTRACT—This paper proposes two kinds of complexity-reduced algorithms for a low density parity check (LDPC) decoder. First, sequential decoding using a partial group is proposed. It has the same hardware complexity and requires a fewer number of iterations with little performance loss. The amount of performance loss can be determined by the designer, based on a tradeoff with the desired reduction in complexity. Second, an early detection method for reducing the computational complexity is proposed. Using a confidence criterion, some bit nodes and check node edges are detected early on during decoding. Once the edges are detected, no further iteration is required; thus early detection reduces the computational complexity.

Keywords—LDPC codes, low complexity, early detect algorithm.

I. Introduction

The high definition television (HDTV) satellite standard, known as the Digital Video Broadcasting (DVB-S2) transmission system, employs a low density parity check (LDPC) coding technique as a channel coding scheme. Unlike turbo codes, LDPC codes have an easily parallelizable decoding algorithm, which consists of simple operations such as addition, comparison, and creation of a look-up table. Moreover, the degree of parallelism is 'adjustable', which makes it easy to trade-off both throughput and complexity. However, the DVB-S2 system requires a large block size and large number of iterations to near Shannon's limit. The standard recommends that the LDPC coded block size be 64800, and the number of iteration be about 70 in the case of a half-coding rate. A large number of iterations for a large block size gives rise to a large number of computation operations, mass power consumption, and decoding delay. It is necessary to reduce the iteration numbers and computation operations without performance degradation in order to implement an LDPC decoder with low power consumption. This paper proposes two kinds of complexity-reduced algorithms. First, sequential decoding with a partial group is proposed. It has the same hardware complexity, and a fewer number of iterations is required at the same level of performance in comparison with a conventional decoder algorithm. The computation of bit node weights and check node weights can seem to be as an approximate projection based on the parity-check matrix; the grouping simply says that this projection can be done in several steps to obtain an approximate answer. Secondly, an early detection method for reducing the computational complexity is proposed. Using a confidence criterion, some bit nodes and check node edges are detected early on during decoding. In this way, because early detected edges are not computed from following iterations, the computational complexity of further processing is reduced.

II. LDPC Decoding Algorithm

The purpose of the decoder is to determine the transmitted values of the bits. Bit nodes and check nodes communicate with each other to accomplish this goal. The decoding starts by assigning the channel values to the outgoing edges, from bit nodes to check nodes. Upon receiving them, the check nodes make use of the parity check equations to update the bit node information and send it back. Each bit node then performs a

Manuscript received Mar. 23, 2005; revised June 19, 2005.

Eun A Choi (phone: +82 42 860 6655, email: eachoi@etri.re.kr) and Deock-Gil Oh (email: dgoh@etri.re.kr) are with Digital Broadcasting Research Division, ETRI, Daejeon, Korea.

Ji-Won Jung (email: jwjung@hanara.kmaritime.ac.kr) is with the Department of Radio and Science Engineering, Korea Maritime University, Busan, Korea.

Nae-Soo Kim (email: nskim@etri.re.kr) is with Telematics/USN Research Division, ETRI, Daejeon, Korea.

soft majority vote among the information reaching him. At this point, if the hard decisions on the bits satisfy all of the parity check equations, it indicates that a valid codeword has been found and the process stops. Otherwise, the bit nodes go on sending the results of their soft majority votes to the check nodes. In the following sections, we describe the decoding algorithm in detail. The number of edges adjacent to a node is called the degree of that node.

1. Initialization

The decoding starts by assigning the channel transmit values to the outgoing edges, from bit nodes to check nodes. The initial channel value is shown in (1).

$$u_n = -L_c \cdot r_n \left(L_c = \frac{2}{\sigma^2}\right), \qquad n = 0,1,\ldots,N-1, \qquad (1)$$

where, N is the codeword size and σ is Gaussian noise variance.

2. Check Node Update

Let us denote the incoming messages to the check node k from its d_c adjacent bit nodes by $v_{n_1\to k}, v_{n_2\to k}, \ldots, v_{n_{dc}\to k}$, as shown in Fig. 1(a). Our aim is to compute the outgoing messages from check node k back to d_c adjacent bit nodes. Let us denote these messages by $w_{k\to n_1}$, $w_{k\to n_2}, \ldots, w_{k\to n_{dc}}$. Each outgoing message from check node k to its adjacent bit nodes is computed as

$$w_{k\to n_i} = g\left(v_{n_1\to k}, v_{n_2\to k}, \ldots, v_{n_{i-1}\to k}, v_{n_{i+1}\to k}, \ldots, v_{n_{dc}\to k}\right),$$
$$g(a,b) = \mathrm{sgn}(a)\times\mathrm{sgn}(b)\times\min(|a|,|b|) + LUT_g(a,b), \text{ and}$$
$$LUT_g(a,b) = \log(1+e^{-|a+b|}) - \log(1+e^{-|a-b|}). \qquad (2)$$

In practice, the $LUT_g(.)$ function is implemented using a small look-up table.

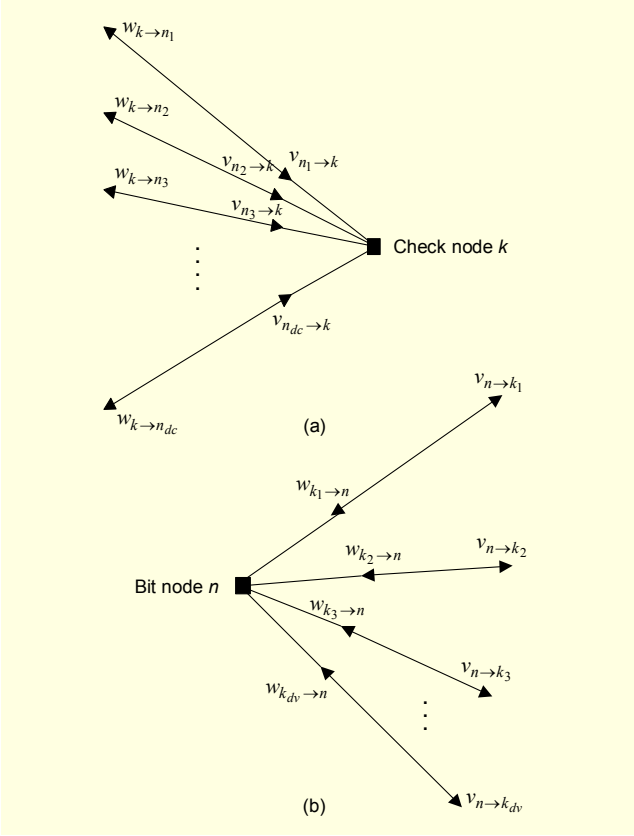

Fig. 1. Message update at (a) check nodes and (b) bit nodes.

3. Bit Node Update

Let us denote the incoming messages to bit node n from its d_v adjacent check nodes by $w_{k_1\to n}$, $w_{k_2\to n}, \ldots, w_{k_{dv}\to n}$, as shown in Fig. 1(b). Our aim is to compute the outgoing messages from bit node n back to d_v adjacent check nodes. Let us denote these messages by $v_{n\to k1}$, $v_{n\to k_2}, \ldots, v_{n\to k_{dv}}$. They are computed as

$$v_{n\to k_i} = u_n + \sum_{j\neq i, j=1}^{dv} w_{k_j\to n}. \qquad (3)$$

Intuitively, this is a soft majority vote on the value of bit n, using all relevant information except $w_{k_j\to n}$.

III. Low Complexity Algorithms

1. Sequential Decoding Algorithm with Partial Group

In this paper, we propose a new decoding structure. First, we divide check nodes by $S_m (m = 1,2,\ldots,p)$ groups. The groups are in general randomly chosen so as to minimize the cycle-4 occurrence. Next, we decode group by group in a serial fashion. For example, Fig. 2 shows the decoding procedure in the case when p equals 2. Check node and bit node probabilities are calculated at the first group, as shown in Fig. 2(a). When the decoding process of the first group has been completed, the second group initiates the decoding process with the calculated bit node probability from the first group, as shown in Fig. 2(b). The proposed algorithm changes only the decoding order without any performance degradation.

If p equals one, this result is the same as a conventional decoder. The resultant extrinsic information of the first decoder is delivered to the second decoder.

The proposed algorithm is based on the same theory. Figure 3 shows the simulation results with codeword size N=64800

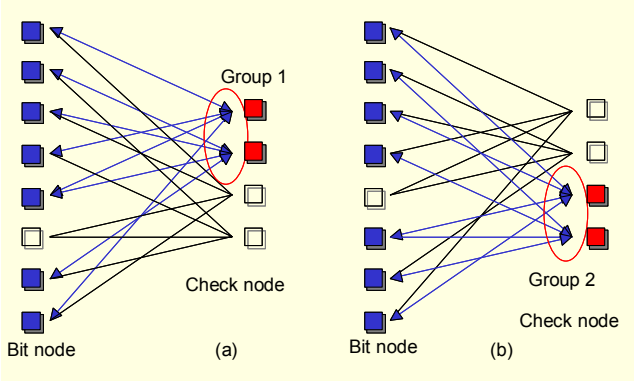

Fig. 2. The bit and check node update procedure (p=2) at (a) the first group and (b) second group.

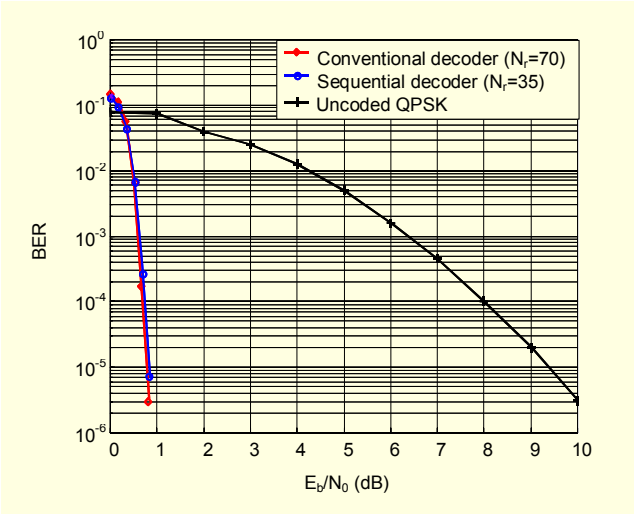

Fig. 3. Bit error rate comparison between conventional LDPC decoder and sequential LDPC decoder (p=2).

and information size K=32400 (number of check node, M=N–K), comparing the bit error rate performances of the conventional and sequential LDPC decoders. With a smaller number of iterations (N_r), the sequential decoding scheme with p = 2 (N_r = 35) shows the performance is almost the same as that of a conventional scheme with p = 1 (N_r = 70).

2. Early Detection Algorithm

Another approach is early detection for reducing the computational complexity. The early detection method is based on the observation that bit nodes and check nodes with a high log-likelihood value can be considered reliable. So, detected bit nodes and check nodes are negligible for the next iteration. This means that we don't need to calculate the bit node and check node updates for a detected node from following iterations. Therefore, computational complexity of the next iterations is reduced. For early detection for the LDPC decoder, the proceedings are as follows:

Step 1. Initialization

$\text{EarlyDetectCheck}[k \to n_i] = 0$

$\text{EarlyDetectBit}[n \to k_i] = 0$

Step 2. Check node updates

if $w_{k \to n_i} \geq T_c$, then $\text{EarlyDetectCheck}[k \to n_i] = 1$

else do (2)

Step 3. Bit node updates

$v_{n \to k_i} \geq T_b$, then $\text{EarlyDetectBit}[n \to k_i] = 1$

else do (3)

If *EarlyDetectCheck*[·] or *EarlyDetectBit*[·] are equal to one, we don't need to calculate the bit node and check node updates, we just deliver the previous values to the bit nodes or check nodes. Therefore, it is very important to choose the early detection threshold: T_b for bit nodes and T_c for check nodes. Fixing on T_c=10, we evaluated the performance for various values of T_b. As shown in Fig. 4, the performance of the early detection method with T_b=18 is the same as that of a conventional scheme.

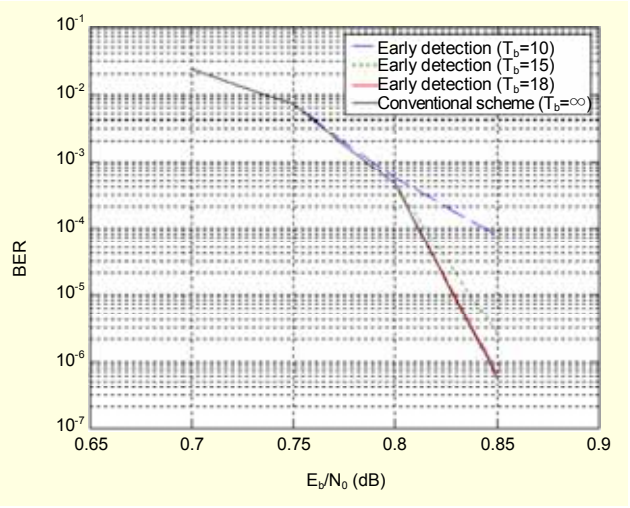

Fig. 4. Bit error rate performance for different T_b (N = 64800, K=32400, d_c = 7, d_v = 13).

The required number of operations associated with the conventional and early detection algorithms are summarized in Table 1, where d_c^* denotes the average number of early detected edges for each check node and N^* denotes the average number of early detected bit nodes. The decoding complexity of an LDPC decoder is evaluated based on (2) and (3).

Fixing on the values of T_b =18 and T_c =10, Table 2 shows the simulated d_c^* and N^* from N_r iterations at E_b/N_0=1 (dB) in the

environment of N=64800, K=32400, M=32400, and d_c=7, d_v=13. As shown in Table 2, the computational complexity of the early detected method is about 50% off in the case of the check node update, and 99% off in the case of the check node update compared to the standard LDPC decoder scheme.

Table 1. Decoding complexities of conventional and early detection methods.

	Conventional decoder	Early detection method
Check nodes ($g(a,b)$)	$(M\times(d_c-1))\times N_r$	$(M\times(d_c-d_c^*-1))\times N_r$
Bit nodes (additions)	$(N\times(d_v-1))\times N_r$	$((N-N^*)\times d_v)\times N_r$

Table 2. The number of early detected edges.

	N_r					
	10	20	30	40	50	60
d_c^*	0.387	1.484	2.619	**3.29**	**3.6**	**3.8**
N^*	491	19048	62372	**64334**	**64335**	**64335**

IV. Conclusions

This paper proposes two kinds of complexity-reduced algorithms for a LDPC decoder. First, sequential decoding with a partial group is proposed. We know that the required number of iterations is almost halved compared with a conventional algorithm maintaining the same computational complexity, and is without performance degradation. Secondly, an early detection method for reducing the computational complexity is proposed. Through Tables 1 and 2, computational complexity of the early-detected method is about 50% off in the case of a check node update, and 99% off in the case of a check node update compared to a conventional scheme. Based on these results, we conclude that the proposed algorithm provides an attractive solution to the implementation of iterative decoding of LDPC codes with large block size and a large number of iterations.

References

[1] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Information Theory*, vol.8, 1962, pp.21-28.

[2] D. J. C. Mackay and R. M. Neal, "Near Shannon Limit Performance of Low-Density Parity-Check Codes," *Electron. Letter*, vol.32, Aug.1996, pp. 1645-1646.

[3] M. Sipser and D. A. Spielman, "Expander Codes," *IEEE Trans. Information Theory*, vol.42, Aug. 1996, pp.1720-1722.

[4] T. Richardson and R. Urbanke, "Efficient Encoding of Low-Density Parity Check Codes," *IEEE Trans. Information Theory*, vol. 47, Feb. 2001, pp. 638-656.

[5] J. W. Bond, S. Hui, and H. Schmidt, "Constructing Low-Density Parity-Check Codes," *Proc. EUROCOMM 2000,* 2000, pp.260-262.

[6] David J. C. Mavkay "Good Error-Correcting Codes Based on Very Sparse Matrics" *IEEE Trans. Information Theory*, vol. 45, no. 2, Mar. 1999.