Fast Convergence Algorithm for Decoding of Low Density Parity Check Codes

Jin Xie*, Liuguo Yin**, Ning Ge*, and Jianhua Lu*
*Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China
**School of Aerospace, Tsinghua University, Beijing, 100084, China
yinlg@tsinghua.edu.cn

Abstract: In this paper, a fast-convergence algorithm is proposed for the decoding of low-density parity-check (LDPC) codes. By putting more weight on the message update from check node to variable node in current iteration, the proposed method speeds up the decoding significantly. Simulation results show that compared with the standard layered min-sum decoding algorithm, the proposed scheme may reduce about 1/6 iteration numbers for LDPC codes used in DVB-S2, with hardware complexity overhead less than 2%.

Key–Words: LDPC code, fast convergence decoding algorithm, layered decoding, min-sum algorithm, hardware implementation

1 Introduction

Due to their outstanding error-correcting capability and inherently parallelizable decoding scheme [1], low-density parity-check (LDPC) codes are adopted by many new generation communication standards, such as DVB-S2 [2], WiMax (IEEE 802.16e) [3], and wireless LAN (IEEE 802.11n) [4]. However, compared with turbo-product codes the decoding of an LDPC code needs much more iterations, which may be a big problem for high throughput decoder design. Hence, it is of great significance to improve the convergence speed of the LDPC decoding.

In recent years, many algorithms have been proposed for the decoding of an LDPC code. The sum-product decoding algorithm may obtain the best decoding performance while its hardware is also the most complex. The min-sum algorithm and its improved versions are good trade-off between decoding performance and hardware complexity. Moreover, these decoding algorithms may be implemented with two schedules. The first one is flooding scheme, which updates all check nodes and variable nodes in two successive steps. The second one is layered schedule [5], which offers much faster convergence speed by means of updating variable node messages sequentially rather than simultaneously [6][7]. Thereby, the min-sum algorithm integrated with layered decoding schedule has been shown to be the best choice for designing LDPC decoders with good decoding performance and low hardware complexity [8][9]. However, since the variable node message update of the layered min-sum algorithm is implemented sequentially rather than simultaneously, there is still a challenging problem for high throughput LDPC decoder design. Many research works have been conducted to further optimize the layered min-sum decoding algorithm to increase the convergence speed. In [10], a fast-convergence decoding method is proposed for LDPC codes used in the WiMax systems. This decoding method serially decodes block codes with identical parity-check matrix, and accelerates the iteration speed by breaking up large codes into small ones. However, this scheme is limited to QC-LDPC codes. In [11], another new decoding algorithm utilizing the zigzag connectivity of linearly encodable LDPC codes is introduced, which is also confined to IRA codes.

In this paper, a fast convergence layered min-sum decoding algorithm is proposed for all kinds of LDPC codes. The idea of this algorithm comes from an observation that during the LDPC decoding, the check-node to variable-node messages computed in current iteration is generally more reliable than those computed in previous iteration. Thus the difference between the check-node to variable-node message from current iteration and that from previous iteration may denote the convergence direction of each variable-node. Based on this observation, we revised the iterative function of the layered min-sum algorithm by slightly amplifying the check-node to variable-node message update. Simulation results show that with the proposed decoding method, the number of iterations may reduce by 8.4% - 62.6% when compared with the decoding with the standard layered min-sum decoding algorithm, which is of great significance for high throughput LDPC decoder design.

The rest of this paper is organized as follows. In section 2, the standard layered min-sum decoding algorithm is described. Then, the improved layered min-sum algorithm is proposed in section 3, and simulation results for the proposed algorithm are presented in section 4. After that, the hardware implementation issue of the proposed method is discussed in section 5. Finally, conclusions are drawn in section 6.

2 LDPC Decoding Schedules

Consider an (N, K) LDPC code defined by a parity check matrix H_{mn}, where N is the codeword length, K is the number of information bits, and $M = N-K$ is the number of parity check euqations. A parity check code can be described by a bipartite graph (Tanner graph [12], see Figure 1) with N variable-nodes corresponding bits in the codeword and M check-nodes representing parity check equations. The edges connecting varible-nodes and check-nodes map the '1's in H_{mn}.

Furthermore, define

$$N(m) = \{n : H_{mn} = 1\}$$

to be the set of variable-nodes connected to check-node m, and K_m to be the number of elements in $N(m)$.

The decoding algorithm exchanges messages between variable-nodes and check-nodes. The messages are posteriori probabilities in the form of Log Likelihood Ratio (LLR), which is defined as:

$$LLR(x_i) = log\{\frac{Pr(x_i = 1)}{Pr(x_i = 0)}\} \quad (1)$$

2.1 Flooding Schedule

Let λ_n^k be the soft output (SO) of bit n, λ_{nm}^k and Λ_{nm}^k be the message sent from variable-node n to check-node m and the message from check-node m to variable-node n in the k^{th} iteration, respectively. The classic flooding schedule updates all the variable-nodes and check-nodes in two successive steps.

During the k^{th} iteration, the check-node m receives the message to its neighboring variable-node n as:

$$\lambda_{nm}^k = \lambda_{ch} + \sum_{n' \in N(m) \backslash n} \Lambda_{nm}^{k-1} \quad (2)$$

where λ_{ch} is the channel a-priori information on the current bit.

Then, combined with the offset min-sum algorithm, updated message from check-node m to its adjacent variable-node n is computed as:

$$\Lambda_{nm}^k = \prod_{n' \in N(m) \backslash n} sign(\lambda_{n'm}^k) * \max(\min_{n' \in N(m) \backslash n} |\lambda_{n'm}^k| - \beta, 0) \quad (3)$$

where β is a positive integer less than 1.

As for the normalized min-sum algorithm, the above updated message is calculated as:

$$\Lambda_{nm}^k = \prod_{n' \in N(m) \backslash n} sign(\lambda_{n'm}^k) * \min_{n' \in N(m) \backslash n} |\lambda_{n'm}^k| / \alpha \quad (4)$$

and α is the normalized factor greater than 1.

Under most circumstances, the offset min-sum algorithm and normalized min-sum algorithm both boost the performance of the min-sum decoder and are the most two widely-used variations of the min-sum algorithm. In some applications, one may precede the other, so we could simulate and compare their performance, thus choosing the better one as the decoding algorithm for specific LDPC codes.

The final decision on a bit is taken in the end of the iteration I. The SO of bit n is calculated as:

$$\lambda_n = \lambda_{ch} + \sum_{n' \in N(m)} \Lambda_{nm}^I \quad (5)$$

If λ_n is negative, the bit n is judged as a 0, otherwise it is considered as a 1.

2.2 Layered Schedule

Unlike the two-phase update in flooding schedule, layered schedule considers the parity check matrix as layers of check equations and updates the variable-node information right after updating check-node information of current layer. Vertical shuffle layer and horizontal shuffle layer have been considered in [13]. In this section, only horizontal shuffle layered schedule is introduced.

The horizontal shuffle decoding is check-node centric, which updates the SO values for each check-node m:

- first, the Λ_{nm}^k are updated using equation (3).
- then, all the λ_n coming out of the variable-nodes connected to this check-node are updated immediately with equation (5).

It is noted that the flooding schedule, the updated messages in the k^{th} iteration are based on the messages calculated in the previous iteration. However, in the layered schedule, this computation is done on messages already updated on the k^{th} iteration by a different check-node. Based on the fact that layered schedule converges faster than the flooding schedule,

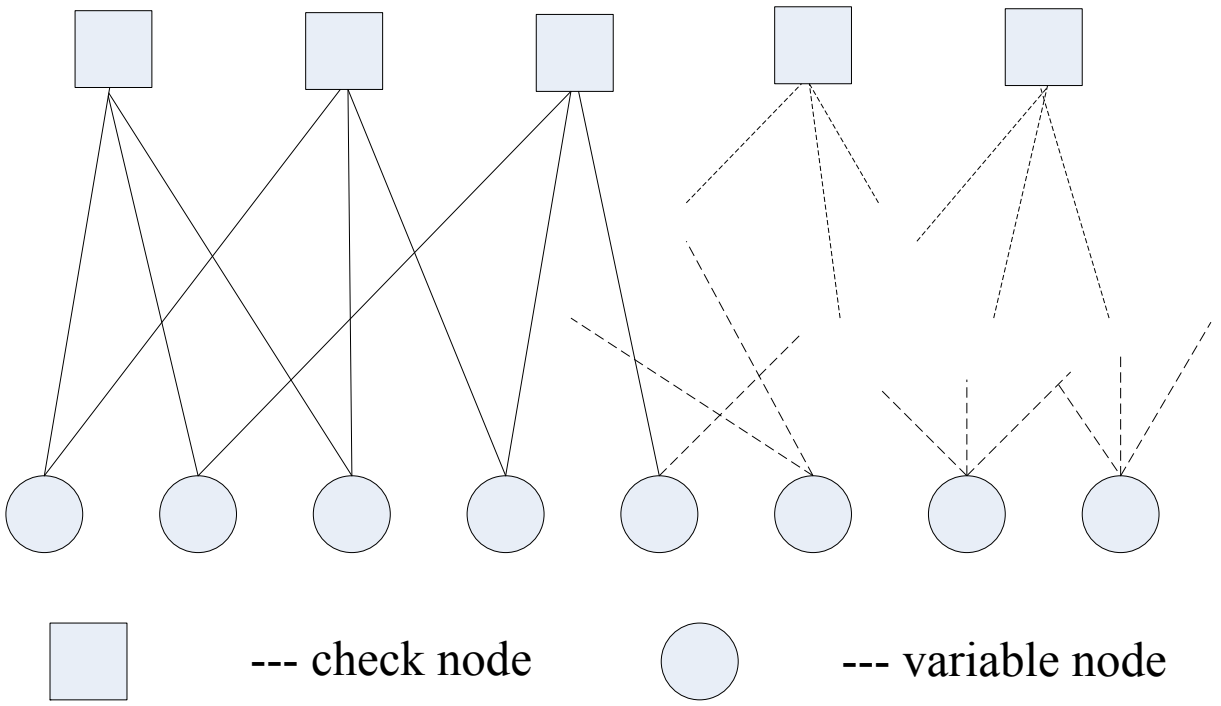

Figure 1: Tanner Graph of LDPC code

we conclude that using messages already updated in the k^{th} iteration, which are statistically more reliable, accelerates the iteration. Further analysis shows that the difference between the k^{th} and $(k-1)^{th}$ messages is $(\Lambda_{nm}^{k} - \Lambda_{nm}^{k-1})$. Adding such a term to the SO value might result in a faster speed than standard layered schedule. Inspired by this, we propose the improved layered decoding as follows.

3 Improved Layered Min-Sum Decoding Algorithm

Similar to the standard layered decoding algorithm, the proposed decoding method can be summarized in three steps:

1. Initialization: A-posteriori information λ_n^0 is initialized with the channel LLR λ_{ch}, and Each Λ_{nm}^0 is initialized to zero. Specifically for BPSK signals transmitted in the AWGN channel,

$$\lambda_{ch} = \frac{2y_n}{\sigma^2}$$

y_n being the the received BPSK signal value, and σ^2 being the noise variance.

2. Iterative decoding: During the k^{th} iteration, we observed that:

$$\lambda_{nm}^{k} = \lambda_n^{k-1} - \Lambda_{nm}^{k-1} \tag{6}$$

As a result of Equ.(6), messages from variable-node to check-node could be generated using the SO and the meomory for λ_{nm}^{k} is saved. When λ_{nm}^{k} is extracted, update messages from variable-node n to check-node m using Eqn.(3).

At the same time, the SO of bit n is computed as:

$$\lambda_n^{int} = \lambda_{nm}^{k} + \Lambda_{nm}^{k} \tag{7}$$

$$\lambda_n^{k} = (1+\omega) * \lambda_n^{int} - \omega * \lambda_n^{k-1} \tag{8}$$

where λ_n^{int} is the interim LLR and ω is the accelerating parameter. If we set $\omega = 0$, λ_n^{int} is equal to λ_n^{k} and our proposed algorithm goes back to the standard layered decoding algorithm.

3. Check stop rules: compute the hard decoding output for current iteration:

$$\hat{c} = \begin{cases} 0 & \text{if } \lambda_n^{k} \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

If constraint function $H \cdot \hat{C} = 0$ is satisfied or the maximum number of iteration has performed, stop the iteration and output decoded results. Otherwise, continue the iteration.

Noted that in Eqn.(8) of the proposed algorithm the variable-node message update function has been

revised by assigning more weight to current LLR . Merging Eqn.(6) and (7) into (8), we get that

$$\lambda_n^k = \lambda_{nm}^k + \Lambda_{nm}^k + \omega * (\Lambda_{nm}^k - \Lambda_{nm}^{k-1}) \qquad (10)$$

Compared with the standard layered decoding, a new term $(\Lambda_{nm}^k - \Lambda_{nm}^{k-1})$ is added. As the check-node to variable-node message becomes more reliable as the iteration proceeds, the term indicates the convergence direction for variable-node n. Therefore, adding such a term to the variable-node message update function will accelerate the convergence speed.

4 Performance Simulations and Numerical Results

4.1 Unconstrained Iteration Number

To evaluate the performance of the proposed improved layered min-sum (ILMS) decoding and the standard layered min-sum (SLMS) decoding, we simulated with DVB-S2 LDPC codes. We choose a maximum of 500 iterations for each decoding, which means approximately unconstrained iteration number. For the min-sum algorithm, $\beta = 0.5$ is used. The parameter ω in Eqn.(8) of the proposed algorithm is set to be 0.05. But further simulations demonstrate that for $0.02 < \omega < 0.06$ the proposed decoding algorithm has almost the same error-correcting performance and convergence speed. Thereby, the performance of the proposed algorithm is not sensitive to parameter ω.

The average numbers of iterations for the two decoding algorithms are listed in Table 1, from which we can see that with the proposed decoding scheme the average number of iterations may be reduced by 8.4%-19.7% at high SNR; while the reduced number of iterations may be increased to be 30% - 62.6% when the channel SNR is low. Moreover, the BER performances with the proposed decoding algorithm and the standard layered min-sum decoding scheme are plotted in Figure 2. It is shown that the proposed decoding method achieves the same error correction performance as the standard layered min-sum decoding scheme. Besides, simulation results are better when the code rate is low, as Table 1 shows that at $1/4$ code rate, the reduced iteration percentage changes slowly when the E_s/N_0 increases, which means our algorithm boosts the iteration significantly for a wider range of SNR.

4.2 Constrained Iteration Number

However, in practical systems, the iteration number is contrained due to throughput requirement. The above simulations set the maximum iteration number to 500, which is approximately the unconstrained situation and is impractical in real-time decoding. So, limited numbers of iteration is considered as follows.

30 iteration numbers are used, for example, in [8][9] to achieve the required performance. However, with the proposed accelerating technique, 25 iteration is enough to achieve the same or even better results. Simulation results is described in Figure 3. We chose $\beta = 0.44$ and $\omega = 1/16$, and simulated 100,000 frames. It is clear that the improved algorithm is better than the standard algorithm for both 30 iteration and 25 iteration. In addition, the improved algorithm with 25 iteration outperforms the standard algorithm with 30 iteration, which is a significant enhancement for real time systems. The average iteration numbers are also listed in table .

4.3 Summary of Simulation Results

It is noted that when the SNR is low, these two algorithms uses the almost the same iteration numbers due to the high error rate. Meanwhile, when the SNR gets higher, the improved algorithm decodes more frames than the standard algorithm and finishes the decoding earlier. It should be pointed out that the simulation programmes for constrained iteration differ from the simulation programmes for unconstrained iteration in the number of parallelly processed layers. while the simulation programmes for unconstrained iteration process one layer each time, the simulation programmes process 90 layers each time in order to reduce the simulation time. The cost is that the parity check matrix is reordered to form a quasi-cyclic matrix. So the results for two programmes are a little different. However, both simulation results show that our proposed algorithm effectively accelerates the standard layered algorithm.

Another important difference between the two simulations is that when the SNR raises, the reduced iteration number increases under the constrained situation, but the reduced iteration number decreases under the unconstrained situation. Because the low SNR leads to high bit error, neither the standard nor the improved algorithm could decode correctly in 30 iteration. So both of the two algorithms ultilize most of the iteration numbers and the reduced iteration number is very few. But when the SNR increses enough, the improved algorithm begins to accelerate the decoding. However, this accelerating effect appears even when the SNR is very low under the unconstrained situation. When the SNR gets too high, both of the stan and the improved algorithm could decode correctly. So the large reduced iteration number narrows with raised SNR.

Table 1: Average Iteration Number of SLMS and ILMS

Code Rate	E_s/N_0(dB)	Average Iteration Number		Average Improvement	
		Standard Layered Decoding	Improved Layered Decoding	Reduced Iteration Number	Reduced Iteration Percentage
1/4	-3.00	117.0	78.7	38.3	32.7%
	-2.95	77.7	56.5	21.2	27.3%
	-2.90	61.7	47.4	14.3	23.2%
	-2.85	51.9	41.7	10.2	19.7%
1/2	0.85	84.7	63.8	20.9	24.7%
	0.90	34.9	28.5	6.4	18.3%
	0.95	25.0	21.8	3.2	12.8%
	1.00	21.6	19.4	2.2	10.2%
2/3	2.95	94.9	35.3	59.6	62.6%
	3.00	33.2	23.3	9.9	29.8%
	3.05	21.3	18.8	2.5	11.7%
	3.10	17.8	16.3	1.5	8.4%

Table 2: Average Iteration Number of SLMS and ILMS for DVB-S2 rate 1/2

Max Iteration	E_s/N_0(dB)	Average Iteration Number		Average Improvement	
		Standard Layered Decoding	Improved Layered Decoding	Reduced Iteration Number	Reduced Iteration Percentage
30	0.8	30.00	29.95	0.05	0.17%
	0.9	29.99	28.23	1.76	5.87%
	1.0	29.42	23.34	6.08	20.67%
	1.1	27.39	19.86	7.53	27.49%
	1.2	25.36	17.67	7.69	30.32%
25	0.8	25.00	25.00	0	0.00%
	0.9	25.00	24.87	0.13	0.52%
	1.0	25.00	22.98	2.02	8.08%
	1.1	24.81	19.86	4.95	19.95%
	1.2	23.95	17.67	6.28	26.22%

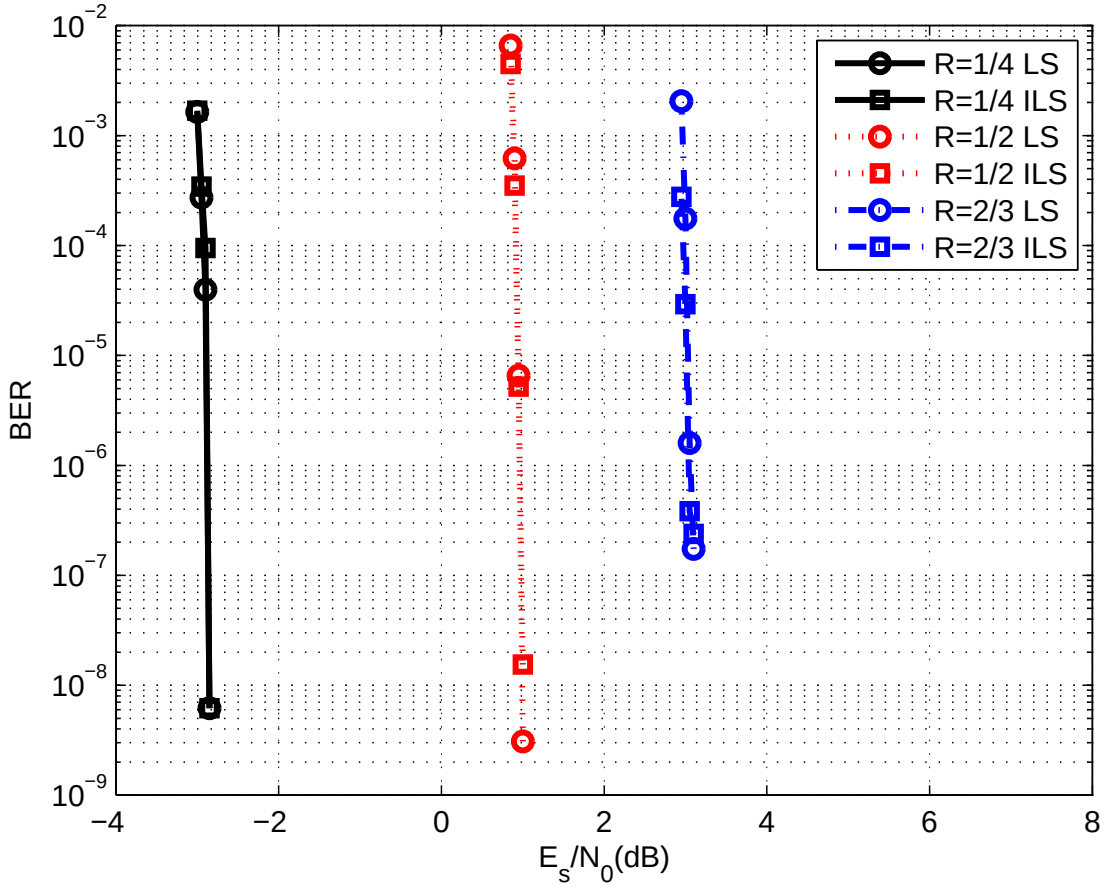

Figure 2: Performance of the ILMS Decoding Algorithm

5 Hardware Implementation Complexity of the Proposed Algorithm

Low-cost hardware architecture for standard layered algorithm has been proposed in many publications [14][15][16][17][18][19]. Most of the effective techniques could be applied to our algorithm. First of all, the check-node message compression technique still works. As mentioned in [8], the magnitudes of all the Λ_{nm}^k from the same check-node m have only two values: either the minimum, or the sub-minimum. So we choose to store the K_m outgoing check-node message signs, 2 magnitudes and the index of the minimum. Second, the quantization and clipping results in [20] also stand for the proposed algorithm because the check-node outgoing messages are the same as the standard layered schedule.

The data path of our proposed algorithm is shown in Figure 4, which is similar to the data path in [8]. It operates in a serial-in, serial-out way, which is convenient for check-node message update and programmable for different codes. In Eqn.(8), λ_n^{k-1} is required to generate λ_n^k. As a result, λ_n^{k-1} is stored into delay registers. The data path decompresses the check-node message on-the-fly, thus effectively reduces the memory especially for high-degree check nodes. In order to implement Eqn.(8) in an economical way, multiplication could be replaced by shift and addition. As mentioned above, our algorithm is insensitve to accelerating parameter ω. For example, ω is set as $1/32$ and the multiplication equals to shift arithmetic right (SAR) 5 bits. As a result, only 3 bits are required to store $\omega * \lambda_n^{k-1}$ if we use 8 bits to quantize λ_n^k. Similarly, $(1+\omega) * \lambda_n^{int}$ is the sum of λ_n^{int} and its SAR.

In many applications, a fully serial solution for LDPC decoder will lead to a low throughput, while a fully parallel solution probably results in congestion in layout due to wire connections between all the check-nodes and variable-nodes. Therefore, for structured LDPC codes like DVB-S2 or WiMax, semi-parallel layered decoding is a useful way to increase

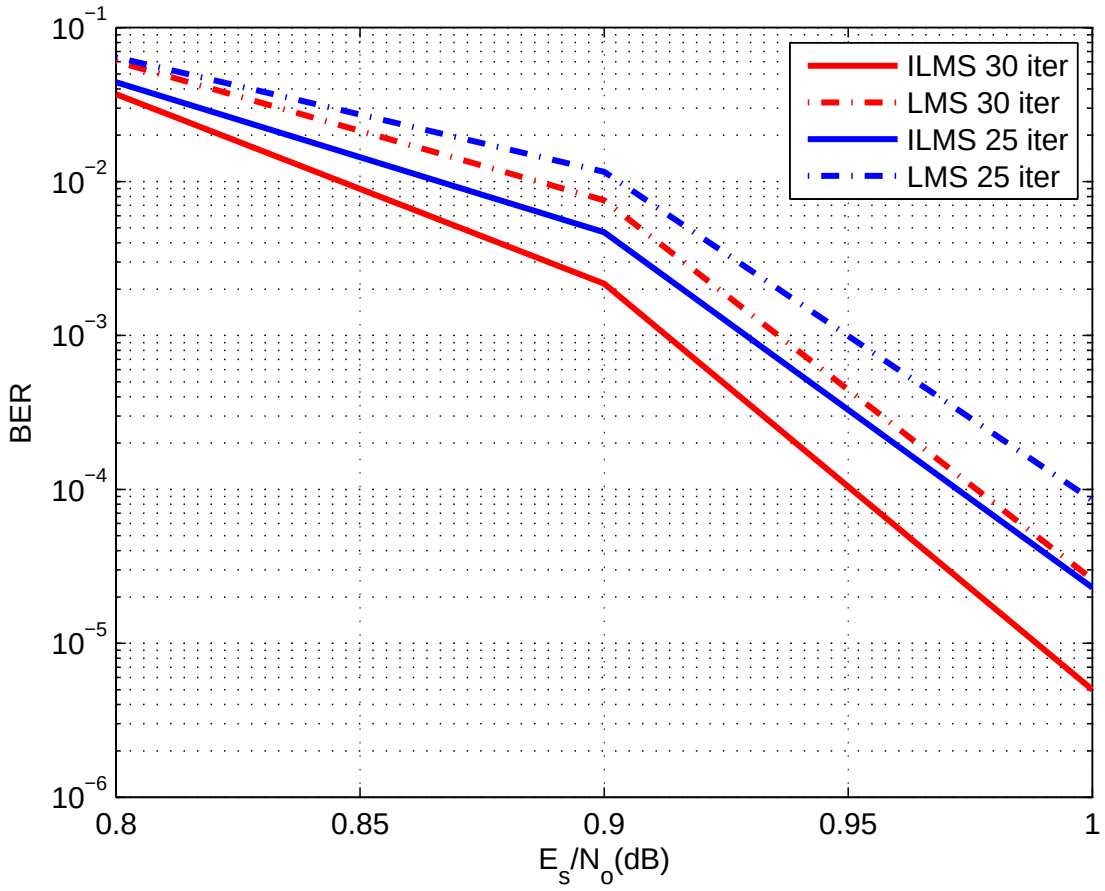

Figure 3: Performance of the ILMS Decoding Algorithm with Limited Iteration Numbers

the throughput of the whole system. Take DVB-S2 for example, there are two layered decoding architectures proposed in [21]. One is the δ-based layered decoding, the other is soft output based layered decoding as shown in Figure 5. In this paper, we discuss the latter because of its simpler implementation. The input parity-check LLRs are stored in an interleaved way so that the corresponding parity check equations are quasi cyclic structured. Blocks of 360*360 submatrix are combination of shifted identity matrix, so the SO values could be read as vectors from the memory and rotated by a barrel shifter to match the order of check equations. The above data path is duplicated to process the incoming SO values simultaneously. After updated by the data paths, the SO values are rotated back and returned to the same position in the SO memory. The addresses of SO values and shift magnitudes are extracted from parity check matrix. The addresses of check-node messages go ahead one step as the data paths have processed current block of parity equations.

The decoders in [6] and [8] use 30 iterations to achieve the BER performance. The proposed architecture, however, reduces 5 iterations, e.g. 25 iterations is enough for the same performance. Thus the throughput increases about 17% with the same clock frequency. At the same time, the cost increases slightly. A 1/2 code rate DVB-S2 decoder is implemented on Xilinx FPGA Virtex 4 LX100. There are 90 parallel data paths in our decoder, and 8 bit inputs and 6 bit message value are used. Consumed resources are listed in Table 3, which depend on the parallel level and quantization bits. Comparing our algorithm with the standard one, only 2% more 4-input LUTs are required and no extra RAMB16s are instantiated. In the case of 45 parallel data paths, synthesis report shows that about 1% more LUTs are required. So approximately the overhead is 1% more LUTs for every 45 data path.

The static timing report shows that a maximum clock frequency of 115.7 MHz is achieved. The 1/2 code rate decoder throughput is 119 Mbps, which is

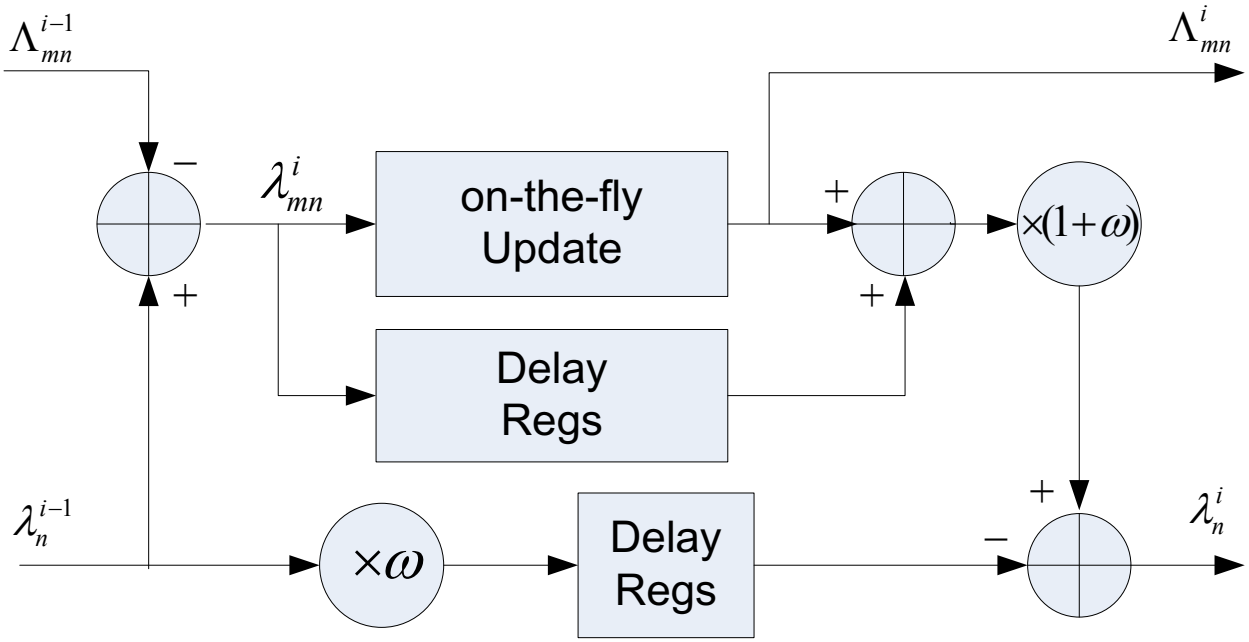

Figure 4: Data Path of the ILMS Decoding Algorithm

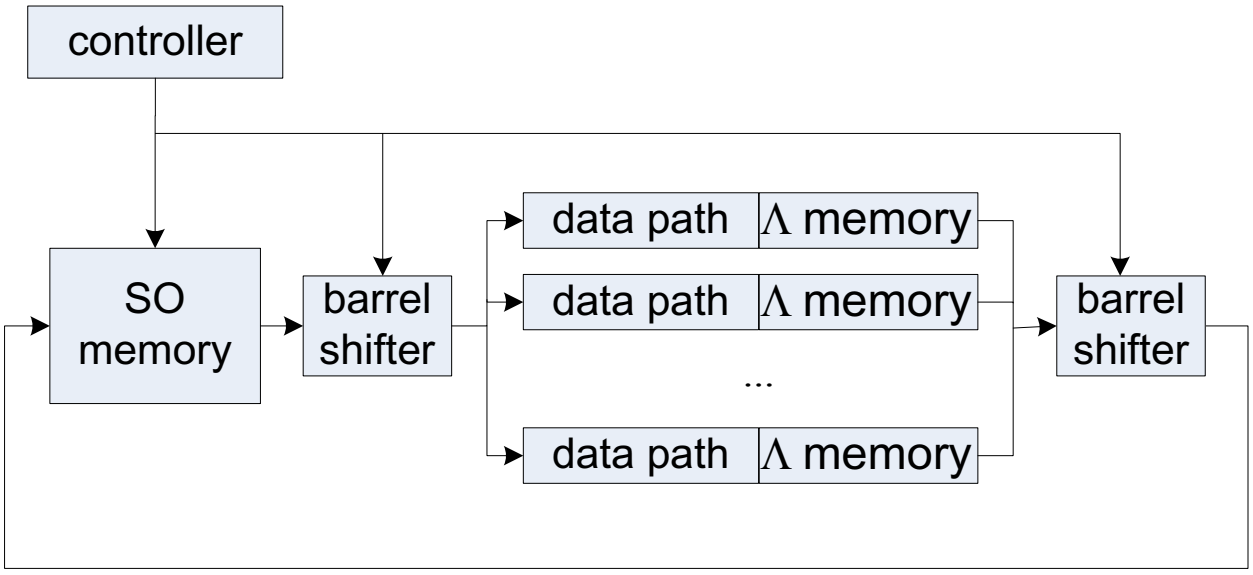

Figure 5: Top Architecture of the ILMS Decoding Algorithm

calculated by:

$$Throughput = \frac{f_{clk} \cdot frame\_length \cdot Ndp}{Niter \cdot K_m \cdot Ncheck} \quad (11)$$

where f_{clk} is the operating clock frequency, Ndp is the number of data path, $Niter$ is the number of iteration, and K_m is a constant for each specific code rate in DVB-S2, $Ncheck$ is the number of check equations.

6 Conclusions

In this paper, we have proposed an improved layered min-sum algorithm for the decoding of low-density parity-check (LDPC) codes. The proposed method speeds up the decoding by slightly amplifying the message update from check node to variable node. It is shown that with our improved scheme the number of iterations may be reduced by 8.4%-62.6%, while the hardware complexity of our method is almost the

Table 3: Synthesis Results On XC4VLX100 FPGA

XST Synthesis Report	ILS	LS
Adders/Subtractors	821	641
Counters	8	8
Registers	9557	7577
Comparators	551	551
Multiplexers	450	360
Xors	720	720
Total Number of 4-LUT	33%	31%
Number of RAMB16s	55%	55%

same as that of the standard min-sum algorithm.

Acknowledgements: This work was supported in part by the National Science Foundation of China (60525107, 60532070) and the National Fundamental Research Program of China (2007CB310600).

References:

[1] D.J.C. MacKay and R.M. Neal, Near Shannon limit performance of low density parity check codes, *Electronics letters.* 33, 1997, pp. 457–458.

[2] European Telecommunications Standards Institude (ETSI), Digital Video Broadcasting (DVB) Second generation framing structure for broadband satellite applications, *EN 302 307 v1.1.1*, 2005.

[3] IEEE 802.16e, Air Interface for Fixed and Mobile Broadband Wireless Access Systems, *IEEE P802.16e/D12 Draft*, 2005.

[4] IEEE 802.11n, Wireless LAN Medium Access Control and Physical Layer specifications: Enhancement for Higher Throughput, *IEEE P802.11n/D1.0*, 2006.

[5] E. Yeo, P. Pakzad, B. Nikolic and V. Anantharam, High Throughput Low-Density Parity-Check Decoder Architectures, *IEEE proceedings of GLOBECOM.* 5, 2001, pp. 3019–3024.

[6] F. Kienle, T. Brack and N. Wehn, A synthesizable IP core for DVB-S2 LDPC code decoding, *IEEE Conference on the Design, Automation and Test in Europe (DATE)*, 2005.

[7] P. Urard, E. Yeo, L. Paumier et.al, A 135Mb/s DVB-S2 Compliant Codec Based on 64800b LDPC and BCH Codes, *ISSCC 2005*, 2005.

[8] J. Dielissen, A. Hekstra and V. Berg, Low cost LDPC decoder for DVB-S2, *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE06)*, 2006, pp. 130-135.

[9] A. Segard, F. Verdier, D. Declercq and P. Urard, A DVB-S2 compliant LDPC decoder integrating the Horizontal Shuffle Scheduling, *Intelligent Signal Processing and Communications*, 2006, pp. 1013–1016.

[10] Y. Ueng and C. Cheng, A Fast-Convergence Decoding Method and Memory-Efficient VLSI Decoder Architecture for Irregular LDPC Codes in the IEEE 802.16e Standards, *IEEE 66th Vehicular Technology Conference*, 2007, pp. 1255–1259.

[11] F. Kienle, T. Lehnigk-Emden and N. Wehn, Fast convergence algorithm for LDPC codes, *IEEE 63rd Vehicular Technology Conference.* 5, 2006.

[12] R. Tanner, A recursive approach to low complexity codes, *IEEE Transactions on Information Theory.* 27, 1981, pp. 533–547.

[13] J. Zhang and M. Fossorier, Shuffled iterative decoding, *IEEE Transactions on Communications.* 53, 2005, pp. 209–213.

[14] E. Choi, D. Chang, D. Oh and J. Jung, Low Computational Complexity Algorithms of LDPC Decoder for DVB-S2 Systems, *IEEE Vehicular Technology Conference* 62, 2005, pp. 536–539.

[15] T. Brack, M. Alles, F. Kienle and N. Wehn, A synthesizable IP core for WIMAX 802.16 E LDPC code decoding, *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2006, pp. 1–5.

[16] T. Brack, M. Alles, T. Lehnigk-Emden, F. Kienle, N. Wehn, N.E. L'Insalata, F. Rossi, M. Rovini and L. Fanucci, Low complexity LDPC code decoders for next generation standards, *Proceedings of the conference on Design, automation and test in Europe*, 2007, pp. 331–336.

[17] S. Jin , G. Minglun, ZH. Zhongjin, L. Li and W. Zhongfeng, A Memory Efficient FPGA Implementation of Quasi-Cyclic LDPC Decoder, *Proceedings of the 5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems*, 2006, pp. 218-223.

[18] W. ANGUS and W.L. LEE, Efficient VLSI Parallel Implementation for LDPC Decoder, *Conference of the 3th WSEAS Int. Conf., Taiwan*, 2004.

[19] W. ANGUS and W.L. LEE, Modified VLSI Implementation for Sequential LDPC Decoder, *Conference of the 3th WSEAS Int. Conf., Taiwan*, 2004.

[20] J. Zhao, F. Zarkeshvari and A. Banihashemi, On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes, *IEEE Transactions on Communications* 53, 2005, pp. 549–554.

[21] M. Rovini, F. Rossi, P. Ciao, N. LInsalata and L. Fanucci, Layered Decoding of Non-Layered LDPC Codes, *Proceedings of the 9th EUROMICRO Conference on Digital System Design*, 2006, pp. 537–544.