HIGH-PERFORMANCE SCHEDULING ALGORITHM FOR PARTIALLY PARALLEL LDPC DECODER

Cheng-Zhou Zhan, Xin-Yu Shih, and An-Yeu(Andy) Wu

Graduate Institute of Electronics Engineering, and
Department of Electrical Engineering, National Taiwan University
Taipei, 106, Taiwan, R.O.C.

ABSTRACT

In this paper, we propose a new scheduling algorithm for the overlapped message passing decoding, which can be applied to general low-density parity check (LDPC) codes. The partially parallel LDPC architecture is commonly used for reducing the area cost of the processing units. The dependency of two kinds of processing units, check node unit (CNU) and bit node unit (BNU), should be considered to enhance the hardware utilization efficiency (HUE). Based on the properties of the parity check matrix of LDPC codes, the updating calculation of the CNU and BNU can be overlapped to reduce the decoding latency by enhancing the HUE with the matrix scheduling algorithm. By applying our proposed LDPC scheduling algorithm to a (1944, 972)-irregular LDPC code, we can get about 60% throughput gain in average without any performance degradation.

Index Terms— LDPC, scheduling, overlapped, matrix, partially-parallel

1. INTRODUCTION

Low-density parity check (LDPC) codes are first introduced by Gallager in 1962 [1] and rediscovered by Dr. MacKay in 1996 [2]. LDPC codes are suitable for high-throughput required applications due to its low decoding complexity and high parallelism. Besides the highly parallel decoding algorithm, LDPC codes also have great error correcting performance close to the Shannon limit [3] within 0.0045dB gap with enough block length.

An LDPC codeword can be iteratively decoded by the updating computing of the check node units (CNU) and bit node units (BNU). The message passing algorithm is applied to the LDPC decoding where the parity check equations and the bit updating equations are computed by CNU and BNU, respectively. The LDPC codes mapped to a fully-parallel structure will generally have higher throughput than the partially parallel architecture, but the area cost of the updating units, CNU and BNU, will also high. Besides the disadvantage of high area cost, the fully parallel architecture of LDPC code will also suffer from the complex routing which will further increase the chip area and causes longer routing paths, and the low hardware utilization efficiency (HUE) which constraints us be not able to put the throughput to the limit. On the other hand, the partially parallel architecture of LDPC codes is adopted for the reason of smaller hardware area but the problem of HUE is still not be solved. The overlapped message passing (OMP) architecture [4] where the updating of CNU and BNU can be partially overlapped is proposed for the partially parallel LDPC architecture to solve the problem of low HUE. Some OMP decoding algorithms are proposed for some specific structure of the LDPC codes [4] - [7], but these algorithms can not be applied to general LDPC codes, while an algorithm for general LDPC codes is also proposed in [8]. In this paper, we propose a higher-performance scheduling algorithm applied for general LDPC codes.

This paper is organized as follows. Section 2 reviews the LDPC codes and the partially parallel architecture. Section 3 briefly reviews the OMP architecture and our proposed scheduling algorithm. The experimental results are shown is Section 4. Finally, Section 5 concludes the paper.

2. GENERAL LDPC DECODER ARCHITECTURE

In this section, basic LDPC decoder concept will be introduced and the architecture of a general LDPC decoder will also be illustrated.

2.1. LDPC Codes

A (*n, j, k*)-regular LDPC codes is defined when LDPC is first introduced by Galleger. The symbols *n, k,* and *j* indicate the column number, column weight, and row weight of the LDPC matrix, respectively. The column weight *k* and the row weight *j* indicate the number of '1' in each column and row of the LDPC matrix. Even the LDPC matrix is a low density matrix, the matrix connection complexity is still considerable in the practical applications. The partially parallel architecture is adopted due to the property of being good at trading off between the throughput and area.

2.2. Partially Parallel Decoder Architecture

The fully parallel LDPC decoder architecture suffers from the high area cost of the processing units due to the updating of each updating requires a CNU or BNU for computing. The numbers of CNU and BNU required for a fully-parallel decoder equal to the numbers of the row and column of the LDPC matrix. In spite of the fully parallel architecture has the advantage of high throughput, it also suffers from the complex routing between the processing units, which will cause larger chip area and longer routing paths. Take a (6, 2, 3)-regular LDPC codes for example. We may use only two check node units and three bit node units for partially parallel LDPC decoding as illustrated in Fig. 1. The terms $m_{a,b}$ indicates the node located at the a-th row and b-th column, and the term I_i is the i-th intrinsic information.

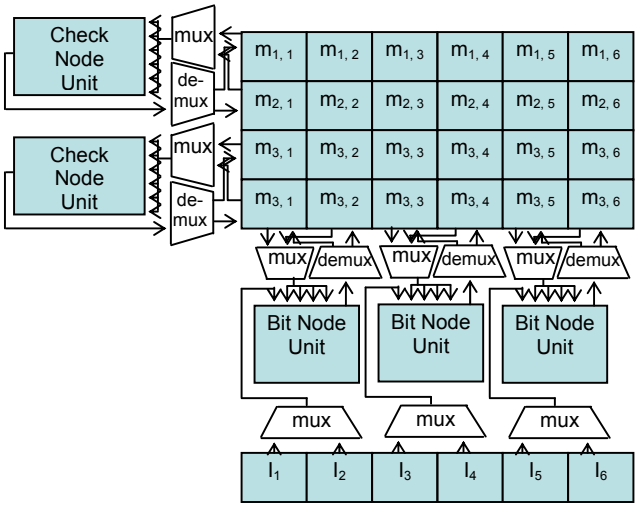


Fig. 1 Partially parallel architecture of LDPC decoder

3. SCHEDULING ALGORITHM OF LDPC MATRIX

In non-overlapped message passing decoding structure, the BNU updating can only be performed after all the CNU updating are done, and the CNU updating can only be executed after all the BNU updating of the last iteration are finished. The updating of CNU and BNU can be partially overlapped after some scheduling algorithms.

3.1. Overlapped Message Passing Algorithm

The overlapped message passing algorithm is first proposed in [4]. We use only one CNU and one BNU here to perform the decoding process to justify the scheduling algorithms. The reason why we just use single CNU and single BNU here is it will be a fair condition to justify if a scheduling algorithm is better than others or not. In other words, if a scheduling algorithm applied to a LDPC matrix under this condition is better than others, it will always have better or equal performance on HUE under any other parallel-processing condition. Assume that we have a LDPC matrix with m row and n columns, and our decoding process needs t iteration to complete the decoding. Fig. 2(a) shows that $(m+n)\times t$ cycles are taken with non-OMP structures, while Fig. 2(b) shows we need only $(n\times t+w)$ cycles to complete the decoding process. The symbol w is the waiting time of BNU at the beginning of the decoding.

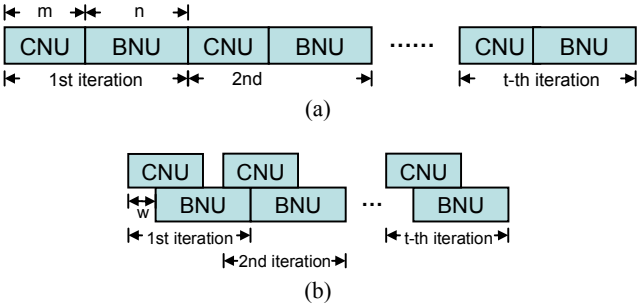


Fig. 2 LDPC decoding (a) without scheduling (b) with scheduling.

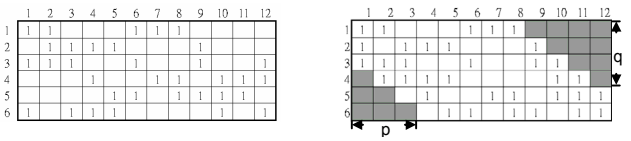
By observing the LDPC matrix, Fig. 3(a) and (b) shows the matrix before and after scheduled by row and column permutation only. We define the row and column number of a LDPC matrix to be m and n, and p and q are the length of the bottom-left and upper-right isosceles right triangle composed of the '0' elements. When the iteration number is large enough for us to ignore the waiting time at the beginning, the decoding throughput gain compared to the straight-forward decoding is

$$throughput-gain \approx \frac{m+n}{(m+n)-(p+q)}, (p+q) \le \min(m,n) \quad (1)$$

$$throughput-gain \approx \frac{m+n}{\max(m,n)}, (p+q) > \min(m,n) \quad (2)$$

Under the condition that we use single CNU and BNU, n is always greater than m. In other applications with more processing units, we can treat m and n as the numbers of total cycles needed to complete all the row and column updating with given CNUs and BNUs, respectively. By equation (1) and (2), we can learn that the best performance of any scheduling algorithm will occur when (p + q) is equal or greater than m. The equation (2) can be also seemed as the limit of a scheduling algorithm when given a fixed number of CNUs and BNUs. In the case of Fig. 3, m = 6, n = 12, p = 3, and q = 4, we can find that $(p+q) > m$. The throughput gain will be

$$throughput-gain \approx \frac{m+n}{\max(m,n)} = \frac{6+12}{12} = 1.5 \quad (3)$$



(a) (b)

Fig. 3 LDPC matrix (a) without scheduling algorithm (b) applied with the scheduling algorithm.

3.2. Our Proposed Scheduling Algorithm

It is almost impossible to find out the optimum schedule with full-search method for a large LDPC matrix. There are several scheduling algorithms proposed for quasi-cyclic LDPC codes [4]-[7], and also some algorithm for general LDPC codes. Compared with [8], our algorithm is more robust, and get a better scheduling performance. The schedule of the LDPC decoding will be arranged before we design our hardware so all scheduling algorithms are off-line computing, and different scheduling algorithms won't cause any extra hardware overhead.

Our proposed scheduling algorithm can be summarized as the flow chart shown in Fig. 4.

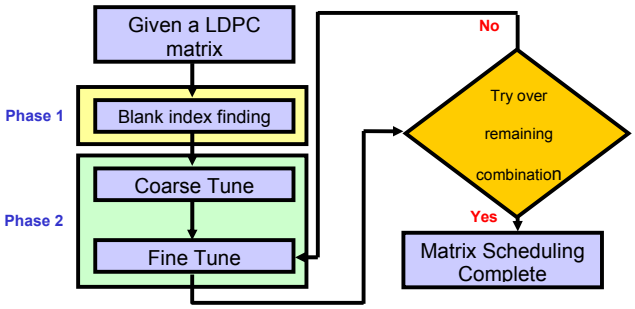


Fig. 4 The flow chart of our LDPC scheduling algorithm.

The scheduling algorithm we propose can be divided into two phases. In the first phase, we try to find out the *blank index* of each '0' element. We first put a '0' element to the top-right corner by matrix row and column permutation. The *blank index* means the side length of the zero isosceles right triangle, which surrounds the '0' element in the top-right corner, we can find also by row and column permutation only. The concept of this phase is similar to that of ref. [8]. The main difference is the algorithm in [8] tries to collect all '1' elements close to the diagonal, while we just try to form a large zero isosceles right triangle in the corner of the matrix in this phase. The steps of the phase 1 are listed as follows, and an example is shown in Fig. 5.

Step1: Put one '0' element to the top-right corner by matrix row and column permutation, and permute all '0's in the same column to make them connected.

Step2: The number of the 0's in other columns, which are in the same rows with that of the most right permuted columns, is called *common zero number* (CZN). If one column has no '0' element in the same row with the corner '0' element, we set the CZN of the column to be 'X' which means we don't care about the CZN of this column. We iteratively put the columns with maximum CZN to the right side of the matrix.

Step3: After Step 2, we can find out a zero isosceles right triangle located at the corner, and the blank index of the corner '0' element can be derived. Then go back to step 1 and find out the blank index of all other zeros. Fig. 6 shows all the blank indexes of each '0' element.

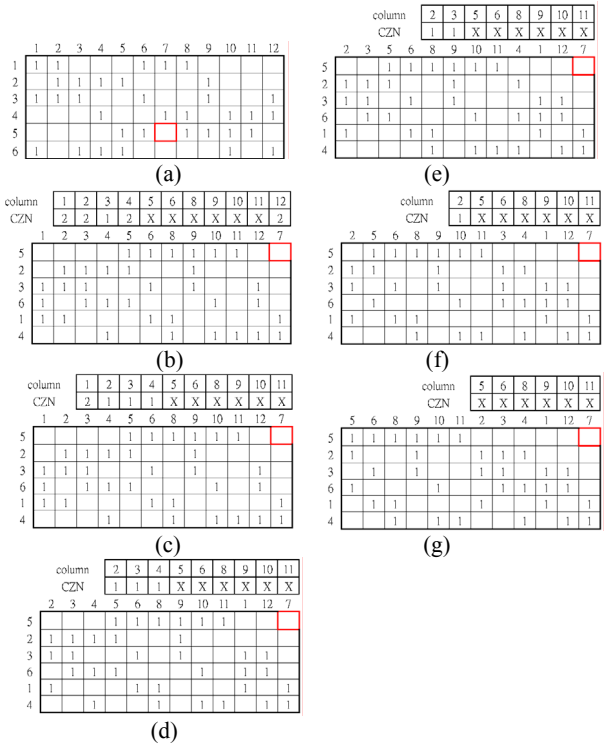


Fig. 5 (a) - (g) show the process of the blank index finding.

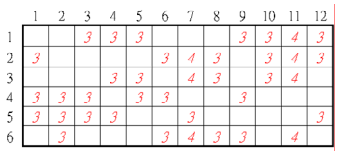


Fig. 6 All the blank indexes of the '0' elements

In phase 2, we have the coarse tune and fine tune stages. The steps of phase 2 are listed below.

Step1: We put the largest isosceles right triangle found in phase 1 to the corner, and set the *free space* region. The free space indicates the region having no rows or columns in common with the zero isosceles right triangle, such as Fig. 7 shows. We can then find the

number of '0' within the free space of each row and denote the number as *zero number* (ZN) in Fig. 7(a).

Step2: Permute the row with greater ZN closer to bottom. In this case, the ZNs of the two rows in the free space are both 5 so permutation is not neededthe rows here.

Step3: Now we check the bottom-zero number (BZN) of the columns related to the free space. The BZN means the number of consecutive '0' elements from bottom up in each column, such as Fig. 7(b). After checking the BZN, we permute the columns from left to right in the descending order of BZN, such as Fig. 7(c). In this example, we have an isosceles right triangle located at the bottom-left corner, whose side length equals to 3. The course tune is done in this step.

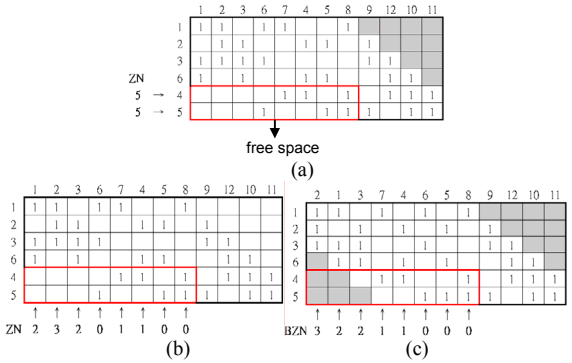


Fig. 7 (a) ZN of each row in the free space
(b) BZN of each column in the free space
(c) The columns after permutation in the descending order of BZN.

From the course tune steps, we found p and q, which indicate the side lengths of bottom-left and top-right triangle, are 3 and 4, respectively. As we have mentioned before, the greater $(p + q)$ is the higher HUE might be reached with the schedule. Because we have know that the value of $(p + q)$ is at least 7 after the course tune and the maximum *blank index* found in phase 1 is 4, the maximum value of $(p + q)$ will never exceed $(4 + 4) = 8$. So we have to only try to find out if there's any permutation leading to $p = q = 4$. The fine tune steps are described as follows.

Step1: Based on the result of $(p + q)$ values derived in the course tune steps, try to find out any possible permutation by using the blank indexes derived in phase 1. In other words, we can iteratively put the '0' elements with maximum blank index to the corner, and try to make large triangles at both corner.

4. EXPERIMENTAL RESULTS

By applying our scheduling algorithm to a 972×1944 LDPC matrix and assume the iteration number is large enough for us to ignore the waiting time at the beginning of the decoding process. We can summarize the performance comparison in Table I. The parallelism (g, h) means we use g CNU and h BNU for partially parallel decoding. It's obvious that the performance of our algorithm is much better than that of the referenced paper. In some situations in the table, the throughput gain is limited to be 1.5 due to the natural restriction described in equation (1) and (2).

Table I Comparison of the throughput gain.

Parallelism	Throughput gain with the algorithm in [8]	Throughput gain with our algorithm
(1, 1)	1.23	1.5
(1, 2)	1.26	1.73
(2, 2)	1.2	1.5
(2, 4)	1.26	1.71
(4, 4)	1.21	1.5
(4, 8)	1.25	1.7

5. CONCLUSIONS

In this paper, we have proposed a new LDPC scheduling algorithm for enhancing the throughput by utilizing the property of the overlapped message passing architecture. The proposed scheduling algorithm in this work is suitable for all kinds of LDPC codes. By applying the scheduling algorithm, we can achieve about 60% higher throughput gain in average compared with non-OMP architecture. The scheduling algorithm for LDPC codes will not cause any performance degradation when decoding.

6. REFERENCES

[1] R. G. Galleger, "Low density parity check codes," *IRE Trans. Info. Theory*, vol. IT-8, pp. 21-28, 1962.

[2] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity check codes," *Electron. Lett.*, vol. 32, p. 1645, 1996.

[3] S. Chung, D. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *IEEE Comm. Letters*, vol. 5, pp. 58-60, Feb. 2001.

[4] Y. Chen, K. K. Parhi, "Overlapped message passing for quasi-cyclic low-density parity check codes," *IEEE Trans. Circuits and Systems*, vol. 51, pp. 1106-1113, Jun. 2004.

[5] Y. Dai and Z. Yan, "Optimal Overlapped Message Passing Decoding for Quasi-Cyclic Low-Density Parity-Check Codes," in *Proc. Global Telecomun. Conf. (Globecom)*, vol. 4, pp. 2395--2399, Dec. 2005.

[6] Daesun Oh, Parhi K.K., "Efficient Highly-Parallel Decoder Architecture for Quasi-Cyclic Low-Density Parity-Check Codes," *IEEE International Symposium on Circuits and Systems*, 2007.

[7] Chen N., Dai Y., Yan Z., "Partly Parallel Overlapped Sum-Product Decoder Architectures for Quasi-Cyclic LDPC Codes," *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp.220-225, Oct. 2006.

[8] I. Park and S. Kang, "Scheduling Algorithm for Partially Parallel Architecture of LDPC Decoder by Matrix Permutation," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 6, pp. 5778--5781, May 2005.