Dual-Rail Decoding of Low-Density Parity-Check Codes

Bongjin Kim, Hasan Ahmed and In-Cheol Park
Dept. of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST)
Daejeon 305-701, Republic of Korea
billyjiny@gmail.com, hasan_153@hotmail.com, icpark@ee.kaist.ac.kr

***Abstract*—In this paper, a new scheduling scheme is proposed to increase the throughput of a low-density parity-check decoder by maximizing resource utilization. The operations of check nodes and variable nodes are fully overlapped in the proposed scheduling to achieve maximized utilization of hardware resources, which in turn increases the throughput and reduces the overall decoding latency. Moreover, no restriction is posed on the formation of the parity check matrix. To verify the effectiveness of the proposed scheme, a series of simulations is performed for irregular random LDPC codes with considering additive white Gaussian noise channel.**

I. INTRODUCTION

Low density parity check (LDPC) codes, characterized by sparse parity check matrices (PCMs), were introduced by Gallager [1] in early 1960s and rediscovered by Mackay [2]. Owing to the excellent error correcting capabilities and the simplicity of iterative decoding, the codes have been adopted in many advanced communication standards such as IEEE 802.11n, 802.16e [3] and DVB-S2 [4]. The belief propagation algorithm [1] is commonly used to iteratively decode the LDPC codes, where two different kinds of messages, check-to-variable (C2V) and variable-to-check (V2C) messages, are computed and exchanged within a single iteration. The messages are passed from one type of nodes to the other type of nodes along the edges of Tanner graph which is a bipartite graph representing the PCM.

A limiting factor in the performance of an LDPC decoder stems from the fact that only one type of updates can be performed at a time in the decoder. The standard schedule, also known as flooding schedule, consists of two phases of message updating; all the check nodes (CNs) are updated first and then all the variable nodes (VNs). Therefore, either CN processers (CNPs) or VN processers (CNPs) are idle during the decoding process, resulting in the limited utilization of hardware resources. A partial solution to this problem is to overlap the operations of CNs and VNs, as proposed in [5] and [6]. The fundamental concept of the overlapped processing is to resolve the dependency between C2V and V2C messages so that some of the node operations can be performed in a parallel or overlapped fashion. The overlapped processing requires a preprocessing on the PCM prior to the actual decoding, but full hardware utilization is not attainable for practical PCMs [5].

The turbo decoding [7] and the shuffled decoding [8], which are the base of the layered decoding [9], were proposed to achieve faster convergence, better error performance and less memory requirement than the flooding schedule, yet the throughput is limited by its multiple sub-iterations. The turbo-sum-product (TSP) scheme proposed recently in [10] can partially achieve the virtues of the shuffled decoding with similar computational complexity compared to the flooding schedule by reducing the number of sub-iterations. However, it has basically no overlap in the node operations and cannot solve the low hardware utilization problem.

In this paper, we present a new scheduling for LDPC decoding. To maximize hardware utilization, the updates of CNs and VNs are fully overlapped and proceed parallely in the proposed scheme, called dual-rail decoding (DRD). Compared to the flooding schedule, the number of cycles taken for an iteration is significantly reduced due to the full overlap of the processing, while the number of iterations is slightly increased in achieving the same error correcting performance. As a result, the proposed DRD is effective in increasing the throughput and shortening the decoding latency. The performance of the DRD scheduling is evaluated through a number of simulations carried out for the rate-1/2 irregular random PCM using the offset min-sum algorithm [11].

II. PREVIOUS WORKS

In the conventional flooding schedule, all the variable nodes are updated after all the check node updating is completed, and vice versa. Let us take the following PCM as an example.

$$PCM = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (1)$$

Fig. 1(a) shows the dataflow graph of the flooding scheduled procedure. The square node represents the row

This work was in part supported by the Korean Research Foundation.

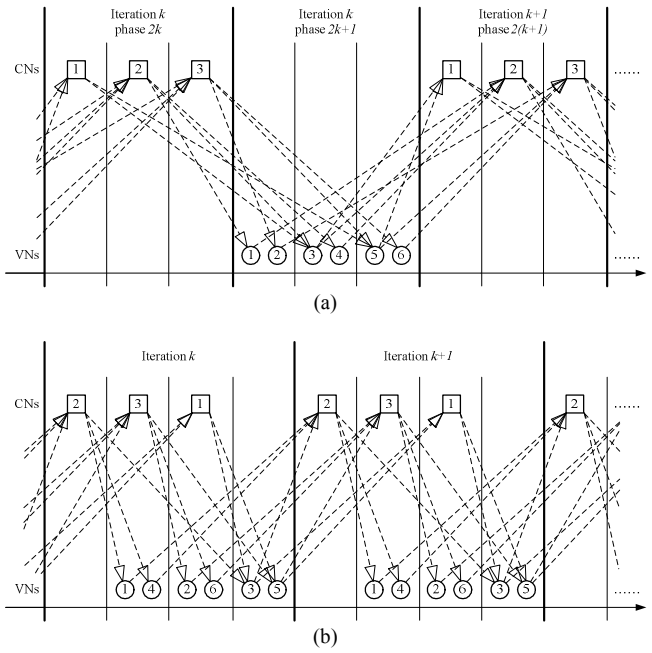

Figure 1. Dataflow graphs for (a) the flooding schedule and (b) the overlapped decoding scheme [5]. The square node represents a row operation performed in a CN and the circular node column operation a VN.

operation of a CN and the circular node represents the column operation of a VN. A thin vertical line shows a clock cycle boundary and a bold vertical line marks a phase boundary. The flooding schedule consists of two separate phases each of which takes 3 clock cycles. As shown in Fig. 1(a), one of the two types of nodes is idle in each phase, simply waiting for the operations of the other type of nodes to be completed.

In the overlapped decoding [5], row and column permutations are applied to the given PCM to resolve the data dependencies so that the node operations can be partially overlapped. Using the algorithm addressed in [5], the permuted version of (1) is obtained as below;

$$PCM = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & 1 & 1 & 0 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 1 \end{bmatrix}. \quad (2)$$

Due to the triangular region, marked with bold 0's in the lower left part, the check node operations and variable node operations can be partially overlapped. The dataflow graph of the overlapped decoding is shown in Fig. 1(b). Since the two phases are overlapped, the phase boundaries are not depicted in Fig. 1(b). After CN 2 is updated, VN 1 and VN 4 can be updated while CN 3 is being updated at the same time, which means that the phases are overlapped resulting in a shorter iteration compared to the conventional flooding schedule. Therefore, 2 clock cycles are overlapped in total, and one iteration is shortened to 4 clock cycles. Although the resource utilization is improved, each type of both nodes has to be idle for one clock cycle in a single iteration.

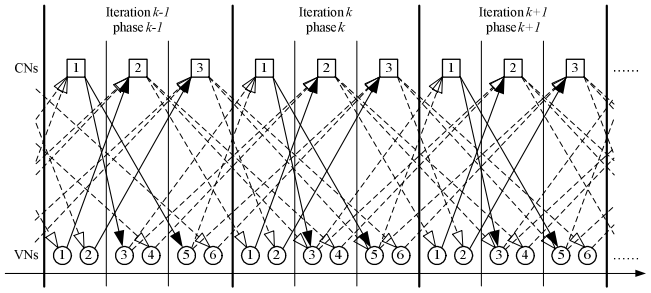

Figure 2. Dataflow graphs the DRD scheme. Solid edges depict intra-phase dataflows, whereas dashed edges represent inter-phase dataflows.

III. DUAL-RAIL DECODING

Fig. 2 illustrates the dataflow graph of the proposed scheduling scheme called dual-rail decoding (DRD). In the figure, the DRD scheme processes one row and two columns at a time regardless of the interconnection of the nodes. Therefore, one iteration is completed in 3 clock cycles and none of the hardware resources stay idle during the whole decoding process. The phrase 'regardless of the interconnection' means that each node does not wait for its neighbors' messages to be ready for use nor any rescheduling is done to resolve the data dependencies. Due to the full overlap, the iteration and the phase boundaries are completely aligned in Fig. 2.

In the decoding process of the DRD, each node processor performs partial update of its messages, which is the same as the shuffled decoding [8]. For example, a CNP computes its outgoing messages by using some inputs updated in the current iteration and others updated in the last iteration. However, unlike the shuffled decoding, the DRD schedules both the CNs and VNs serially.

In order to clarify how the proposed DRD scheme works, we define two different kinds of dataflows in Fig. 2. The dataflow of a message that can be computed and utilized in the same phase is called *intra-phase* dataflow, whereas the dataflow of a message that is generated in the previous phase and utilized in the current phase is called *inter-phase* dataflow. In Fig. 2, intra-phase dataflows are denoted by solid lines, and inter-phase dataflows are denoted by dashed lines. For example, the intra-phase edge between VN 1 and CN 2 indicates that the message updated by VN 1 is quickly utilized by CN 2 in the same phase. As CN 3 and VN 6 are processed in the same cycle, the message updated by CN 3 cannot be utilized by VN 6 immediately, and therefore, an inter-phase dataflow is depicted between CN 3 and VN 6.

Every undirected two-way edge in the Tanner graph is decomposed into two directed edges in order to clearly represent the dataflow type. Note that, except the edges between the CNs and VNs processed in the same cycle, such as CN 3 and VN 6 in Fig. 1(b), every edge in the Tanner graph can be decomposed into one inter-phase dataflow and one intra-phase dataflow. It is clear that the intra-phase dataflows shorten the decoding latency, and ensure the quick utilization of updated messages which in turn increase the rate of convergence.

As intra-phase and inter-phase dataflows are conceptual distinction for analysis, we do not need to take into account the dataflow type in the hardware implementation. To process a node, it is sufficient to access the neighbor messages available at the time. If the message was calculated in the previous phase, the dataflow is classified as an inter-phase dataflow. Otherwise, it is classified as an intra-phase dataflow.

For a given $M \times N$ PCM, the number of the check node units and the variable node units instantiated in the hardware is denoted as *NC* and *NV*, respectively. As *NC* rows and *NV* columns are processed in parallel in each clock cycle, a single iteration can be completed in total of $max(\lceil M/NC \rceil, \lceil N/NV \rceil)$ clock cycles. Clearly, the hardware utilization can be maximized when $M/NC = N/NV$ and $M\%NC = N\%NV = 0$, where $\%$ is the remainder operator.

Compared to the flooding schedule, the proposed method takes much less time to process a single iteration, as all the hardware resources are concurrently utilized without resorting to the data dependencies between CNs and VNs. Therefore, the proposed DRD does not necessitate any preprocessing such as column or row permutation of the given PCM which is unavoidable in the overlapped scheme [5].

The DRD approach does not provide any reduction in the number of iterations, due to the partial update of the messages. The number of iterations may slightly increase and cause the degradation in error performance under a fixed number of maximum allowed iterations (MAI). This can be regarded as a drawback since the shuffled decoding can reduce the number of iterations compared to the flooding schedule. However, the term 'iteration' in the DRD is different from that of the shuffled decoding. In the shuffled decoding, each column operation includes its own phase of VN operations, which means one iteration actually can be seen as a super-iteration associated with multiple sub-iterations. Therefore, in an architecture where a CN takes the input messages in parallel, due to the more complex operation, the clock cycle must have longer duration. Otherwise, a CN can take the input serially which will require much more cycles for an iteration. The DRD, on the other hand, overlaps one CN phase and one VN phase without any sub-iteration, resulting in shorter processing latency and lower computational complexity compared to the shuffled decoding. As the DRD decoder has shorter latency than the flooding decoder, we can achieve better performance by increasing the MAI.

The error correcting performance of the DRD can be further improved by increasing the intra-phase dataflows, which can be achieved by permuting the columns of the PCM. The objective of the column permutation is to reduce the inter-phase dataflows caused by the CNs and VNs processed at the same cycle, and thus, the desired PCM should be empty in the diagonal direction. The column permutation algorithm is briefly outlined below with assuming $NC = 1$ and $NV = 2$, and it can be easily generalized for arbitrary *NC* and *NV*. The rate-1/2 PCM defined in IEEE 802.16e [3] is taken for an example and the proposed column permutation algorithm is applied. The result of the permutation is shown in Fig. 3, where it is assumed that one block row and two block columns are processed at a cycle. In the algorithm, the candidate columns

5	7	2	9	0	8	11	14	18	21	1	3	20	4	22	23	10	6	12	16	15	17	19	13
		73	83		55					94								7					0
22	9					12	0			27							79						0
81	33					0	0				24		22							0			
		47	25	61	65														0	0			
		39	41													72	84		0		0		
40	82					73		0					46					0			0		
		95	14					0			53					18						0	
		73	47							11		0					2					0	
24	43			12		51			0			0	83										
94	59					72			0					0		70							
		7	49		39						65			0	0								
66	41			43		26									0			7					

Figure 3. The rate-1/2 base PCM defined in IEEE 802.16e [3] is column permuted by the proposed algorithm. The indices above the PCM denote the original column indices.

of a certain row indicate the columns which are empty in the corresponding row position, and vice versa.

Step 1) Count the number of candidate columns for each row and choose the row with least number of candidates.

Step 2) Count the number of candidate rows for each candidate column of the chosen row. Select two columns that have least numbers of candidates and assign them to the row.

Step 3) If there are remaining rows and columns, go to Step 1. Otherwise reorder the determined pairs in proper way so that the reordered PCM will be empty in the diagonal direction.

IV. Simulation Results

For the rate-1/2 irregular (2000, 1000) PCM constructed using Neal's software [12], the performance of the proposed DRD scheme is evaluated with assuming additive white Gaussian noise channel. All simulations use the offset min-sum algorithm [11].

Fig. 4 compares the performance of the DRD to that of the flooding schedule under varying SNR and fixed MAI of 15 in terms of bit error rate (BER) and frame error rate (FER). The error performance of the DRD with *NC* of 1 shows negligible degradation compared to that of the flooding schedule throughout the entire SNR region. As *NC* increases, the FER performance degrades slightly, because the number of inter-phase dataflows increases according to *NC*. Though more iterations are needed to achieve the same level of error correcting performance, the number of additional iterations is small. Therefore, the DRD provides a good tradeoff between the error performance and the number of clock cycles. In addition, Fig. 4 shows that using the column permuted PCM can improve the error performance, as discussed in Section III.

Assuming $M \times N$ PCM, Table I summarizes the hardware utilization (HU) and cycles per iteration (CPI) of the proposed DRD and other scheduling schemes. In the CPI equation, w_r denotes the weight of a row in the PCM and $\max(w_r)$ denotes the maximum weight among the w_r's. For the simplicity, it is assumed that overlapped scheme reorganizes the PCM so that each type of nodes stays idle for 25% time of

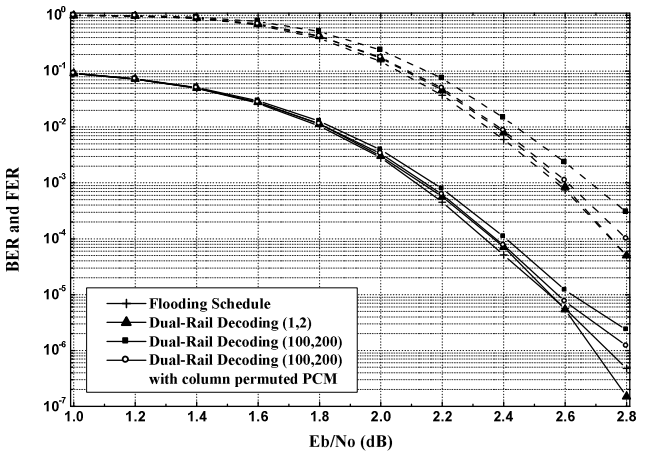

Figure 4. Comparison of the BER and FER resulting from the flooding schedule and the DRD (solid line: BER; dash line: FER). The result of using the column permuted PCM is included.

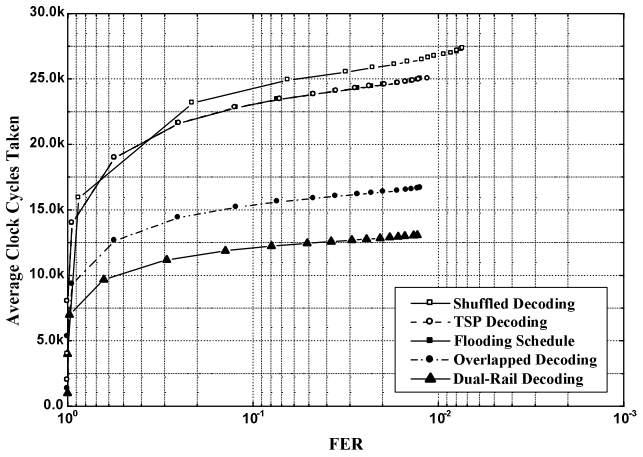

Figure 5. Comparison of the average clock cycles taken by various decoding schedules under a fixed SNR of 2.0dB.

TABLE I. COMPARISON OF HARDWARE UTILIZATION AND CYCLES PER ITERATION

Scheduling	HU[a]	CPI[b]
Proposed	100%	$\max(\lceil M/NC \rceil, \lceil N/NV \rceil)$
Flooding	50%	$\lceil M/NC \rceil + \lceil N/NV \rceil$
Overlapped [5]	75%	$(\lceil M/NC \rceil + \lceil N/NV \rceil) \times 2/3$
Shuffled [8]	50%	$\lceil M/NC \rceil + \Sigma \lceil w_r/NV \rceil$
TSP [10]	50%	$\lceil M/NC \rceil + \lceil N/NV \rceil + \lceil \max(w_r)/NV \rceil$

a. Hardware utilization

b. Cycles per iteration

an iteration. The HU of the shuffled decoding and the TSP scheme are both 50% since no overlapping is allowed in the node operations. Furthermore, the numbers of CPI of the two schemes are larger than that of the flooding schedule due to the sub-iterations. The shuffled decoding performs VN operations by the number of nonzero entries in the PCM, whereas the TSP requires additional VN operations on the row which has largest number of nonzero entries.

Fig. 5 shows the average number of clock cycles required to achieve a certain FER, where the DRD takes much less cycles than other schemes even though it requires a few more iterations. Given a fixed amount of hardware resources, hence, the proposed DRD provides higher throughput than the conventional approaches do.

V. CONCLUSION

In this paper, we have proposed a novel scheduling scheme, named dual-rail decoding, which maximizes the hardware utilization of the LDPC decoder. The DRD scheme performs the row and column operations in parallel regardless of their data dependencies so as to reduce the idle time of the CNPs and VNPs and increase the throughput of the decoder. Compared to the flooding schedule, the overlapped scheme, and the shuffled decoding, the proposed DRD increases the number of iterations slightly, but it reduces the number of cycles taken for a single iteration. As a result, the DRD requires much less cycles in achieving a certain FER performance.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," IRE *Trans. Inf. Theory*, vol. 8, pp. 21-28, Jan. 1962.

[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399-431, Mar. 1999.

[3] IEEE 802.16e. Air interface for fixed and mobile broadband wireless access systems. IEEE 802.16e/D12 Draft, Oct. 2005.

[4] Digital video broadcasting (DVB); second generation. ETSI EN 302 307 v1.1.1, 2005.

[5] S. H. Kang and I. C. Park, "Loosely coupled memory-based decoding architecture for low density parity check codes," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 5, pp. 1045-1056, May 2006.

[6] Y. Chen and K. K. Parhi, "Overlapped message passing for quasi-cyclic low-density parity check codes," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 6, pp. 1106-1113, June 2004.

[7] M. M. Mansour and N. R. Shanbhag, "High throughput LDPC decoders," *IEEE Trans. VLSI Sytems*, vol. 11, pp. 976-996, Dec. 2003.

[8] J. Zhang and M. P. C. Fossorier, "Shuffled belief propagation decoding," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, vol. 1, pp. 8-15, Nov. 2002.

[9] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop Signal Processing and Systems (SiPS. 04)*, Austin, TX, pp. 107-112, Oct. 2004.

[10] Y. Dai, Z. Yan, and N. Chen, "Memory-efficient and high-throughput decoding of quasi-cyclic LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 4, pp. 879-883, April 2009.

[11] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier and X. Hu, "Reduced-Complexity Decoding of LDPC Codes", *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288-1299, Aug. 2005.

[12] Software for low density parity check (LDPC) codes, R. M. Neal. [Online]. Available: http://www.cs.utoronto.ca/~radford/ldpc.software.html.