Flexible LDPC Decoder Architecture for High-Throughput Applications

Sangmin Kim and Gerald E. Sobelman
Department of Electrical & Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA
{sangmin, sobelman}@ece.umn.edu

Hanho Lee
School of Information & Communication Engineering
Inha University, Incheon, 402-751, Korea
hhlee@inha.ac.kr

Abstract—In this paper, we present a flexible high-throughput LDPC decoder architecture that can support different code rates and block sizes in wireless applications such as IEEE 802.11n, IEEE 802.16e, and IEEE 802.15.3c standards. Several flexible LDPC decoders have been presented in the literature but their throughput (less than 640 Mbps) is limited due to block-serial scheduling of the decoding computations. The proposed architecture is based on a block-parallel scheduling scheme using a layered decoding method. To achieve higher throughput, check node-based processes are implemented in a fully parallel architecture and the memory is partitioned into a number of banks. System flexibility is achieved by allowing the check node-based units and the memory banks to be configured according to the code rate and block size of the LDPC code of interest.

I. INTRODUCTION

Low Density Parity Check (LDPC) codes have attracted much attention because of their excellent error correcting performance and inherently parallelizable VLSI implementation. Therefore, they are being widely used in communication standards such as DVB-S2, IEEE 802.16e and IEEE 802.11n. In addition, mmWave (millimeter wave) Wireless Personal Area Networks (WPANs) described by the IEEE 802.15.3c Working Group [1] are considering LDPC codes as the preferred choice for forward error correction (FEC). All of the LDPC codes in the above standards are based on “Architecture-Aware LDPC codes” [2] or “Block-LDPC codes” [3]. The parity-check matrix H of these codes is partitioned into block-columns and block-rows, which are particularly suitable for VLSI implementations by simplifying the memory access and utilizing a well developed switching network.

LDPC codes are decoded using an iterative message-passing algorithm, consisting of a row operation and a column operation (called the two-phase message passing algorithm), over a graph-based representation of the codes. A method of determining the update order between the row operations and the column operations is called a scheduling. Various scheduling schemes have been proposed, such as a flooding schedule and a serial or layered schedule [4]. The flooding schedule updates all row operations after updating all column operations and vice versa, while the layered schedule updates the row (column) operations by sending an immediately updated column (row) message. The layered decoding schedule can reduce the number of iterations by almost 50% without performance degradation compared to the flooding decoding schedule. In other words, it achieves approximately twice as fast decoding convergence due to the use of intermediate check-node (or variable-node) message values. A low complexity LDPC decoder architecture using the layered decoding schedule was developed in [5].

A semi-parallel or block-serial architecture of a layered LDPC decoder has been presented in the literature [6]-[8] to increase the convergence speed and to reduce latency. However, it has low decoder throughput due to its block-serial scheduling architecture. In this paper, we propose a check node-based processor (CNBP) architecture suitable for improving decoding throughput while achieving system flexibility, which is necessary for next-generation mobile communication systems. A novel architecture based on block-parallel operations (simultaneously processed group by group) using a layered decoding schedule is developed that uses parallel memory accesses. The rest of the paper is organized as follows. In Section II, we provide the background for block-LDPC codes and the layered decoding schedule. In Section III, we propose a block-parallel LDPC decoder based on a novel architecture for the check node-based processor. In addition, system flexibility will be described which allows reconfiguration of the LDPC decoder. Finally, our conclusions are presented in Section IV.

II. REVIEW OF LAYERED DECODING SCHEDULE

After giving a brief introduction to Block-LDPC codes, the layered decoding schedule is addressed in this section. The layered decoding schedule allows the use of efficient block-serial decoder architectures. Although the block-serial decoder architecture is efficient for achieving system...
flexibility, its throughput is limited due to the serial architecture of the message processing units. For example, a multi-edge type vector LDPC decoder, as proposed by Richardson [9], can be implemented at low hardware complexity but it has a relatively low decoder throughput.

A. Block-LDPC Codes

Block-LDPC codes described in several IEEE standards have constraints or structures which can be exploited in implementing both the encoder and decoder. For example, the IEEE 802.15.3c LDPC codes shown in [1] consist of blocks with different cyclic shifts, and can support very low complexity systematic encoders and low complexity, highly parallelizable decoders. The \( M_b \times N_b \) base matrix \( H_b \) with \( M_b = M/2 \) and \( N_b = N/z \), where \( M \) is the number of parity check equations, \( N \) is the code length, and \( z \) is the sub-matrix size, in the IEEE 802.15.3c LDPC codes have 32 columns of blocks and \((1-\text{code rate}) \times 32 \) rows of blocks. For example, at rate \( \frac{1}{2} \) for \( N = 672 \), the parity check matrix has \( M_b = 16 \) layers, the size of the permutation sub-matrix is \( z = 21 \), and the column weight of each layer is at most 1. An example of the \( 4 \times 5 \) base matrix \( H_b \) is shown in Fig. 1.

B. Layered Decoding Schedule

A brief overview of the decoding algorithm is provided to describe the layered decoding scheme and the architectural issues.

In the iterative message passing algorithm, messages are denoted by \( R_v \) (row operation) for extrinsic messages from the check node \( c \) to the variable node \( v \) and by \( L_v \) (column operation) for extrinsic messages from the variable node \( v \) to the check node \( c \). The update operations at the check nodes and variable nodes can be expressed as in equations (1) and (2), respectively.

\[
R_v = - \prod_{m \in N(c)} \text{sign}(L_{mc}) \cdot \Psi(v) \sum_{m \in M(v)} \Psi(L_{mv}) \tag{1}
\]

\[
L_v = \sum_{m \in M(v)} R_{mv} - y_v \tag{2}
\]

\( N(c) \) and \( M(v) \) denote the set of positions of the columns of \( H \) and the set of position of the rows of \( H \) such that, \( N(c) = \{v | H_{vc} = 1\} \) and \( M(v) = \{c | H_{cv} = 1\} \), respectively. In equation (2), the \( \Psi \)-function, \( \Psi(x) = \log(\tanh(x/2)) \), is a nonlinear function and \( y_v \) is the prior log-likelihood ratio (LLR) given by \( 2r_v/\sigma^2 \), where \( r_v \) is the AWGN channel output and \( \sigma^2 \) is the noise variance. At every iteration (one iteration consists of equations (1) and (2)), the soft decoding result for each bit is determined as follows:

\[
L_v = \sum_{c \in M(v)} R_{cv} - y_v \tag{3}
\]

In contrast to the iterative message passing algorithm using a flooding schedule, the layered decoding schedule processes the \( M_b \)th row (or \( N_b \)th column) of \( H \) in layers or groups. In our work, we use a horizontal layer decoding scheme for application to a check node-based processor. For each variable node \( v \) inside the current \( M_b \)th row, \( R_v \) in equation (1) is computed and is immediately used for the next layer. Instead of using \( L_v \) messages, variable node messages for each column block are used to update the \( R_v \) messages on the fly, thus avoiding the need to maintain additional memory for the \( L_v \) messages. A more detailed description of the layered decoding schedule is given in [4] and [5]. We propose a block-parallel LDPC decoder by reformulating the check node-based computation of the horizontal layered decoding schedule for improved throughput, as will be discussed in the next section.

III. FLEXIBLE LDPC DECODER ARCHITECTURE

The proposed decoder architecture is based on a multi-edge type vector LDPC decoder [9], but it has been reformulated to increase the throughput and to achieve system flexibility. In [9], a vector of \( z \) processors (\( z \) check/variable node processors) operates on a macro column/row sequentially with one sub-matrix. For instance, the block-serial decoder needs at least the maximum check node degree \( (d_c = 3) \) number of clock cycles to process three messages (m1, m4, m7), as shown in Fig. 2. In the proposed architecture, all messages (m1, m4, m7) can be simultaneously processed in a single clock cycle, which will considerably improve the throughput of the decoder.
As depicted in Fig. 3, the proposed block-parallel LDPC decoder mainly consists of two memory blocks for storing messages, check node-based processors (CNBPs) for processing intermediate messages, switching networks (SNs) for routing messages, a parity check module and a decoder control module. Fig. 4 shows an example processing for the first row \((m_1, m_4, m_7)\) of a \(4\times5\) \(H_b\) matrix, showing the relationship between messages and memory blocks through the SNs. The architecture of a CNBP suggests the use of parallel structures for achieving faster decoding convergence of the layered decoding schedule. Let \(m(i)\) \((i = 1, 2, \ldots, 7)\) represent the \(i\)-th element of each message vector. For each element \(i\) of each message vector per row block, the number of inputs in the CNBP depends on the value of \(d_c\). A variable node memory (VN_MEM) block includes \(N_v = d_c \times K\) bit values with \(K\)-bit precision corresponding to one edge. The VN_MEM block \(N_v\)'s are used to read/write variable-to-check messages while the CNBP performs the block row \((m_1(i), m_4(i), m_7(i))\) processes shown in Fig. 3. In other words, the CNBP simultaneously processes several block edges adjacent to the \(M_{dc}\) block check node. A check-to-variable memory (C2V_MEM) block stores \(L \times z \times K\) bit values, where \(L\) indicates the number of non-zero integers in the base matrix \(H_b\). The C2V_MEM block is partitioned into \(d_v\) banks. A C2V_MEM bank address selects sets of the \(d_v\) banks to be read or written.

A switch network (SN) that implements rotations of the input message vector is available in the Benes [10] network. In our work, \(2 \times d_v\) SNs are required for switching message outputs from the CNBP to the VN_MEM, and for switching messages output from the VN_MEM to the CNBP. In addition, a specific memory that is responsible for storing pre-computed routing patterns should be able to provide for different code rates and block sizes.

In order to clarify the higher throughput provided by our proposed block-parallel LDPC decoder, the throughput of the decoder can be estimated as:

As shown in Fig. 5, the throughput of the proposed block-parallel LDPC decoder can be estimated as:

\[
\text{Throughput} = R \times N \times \frac{f_{clk_{\max}}}{\text{iterations}} \times M_{dc}
\]

where \(R\) is the code rate, \(f_{clk_{\max}}\) is the maximal clock frequency, and \(M_{dc}\) is the number of block rows corresponding to \(R\). This approximate throughput is not related to the total number of message edges, \(L\), whereas the throughput in a block-serial decoder depends on \(L\).

For implementing the CNBP in a fully parallel architecture, the layered decoding schedule [4] could be equivalently reformulated as follows.

Initialization: \(R_s = 0\), \(\forall c\) and \(\forall v \in N(c)\)

\[
Q_c = y_c + \sum_{\alpha \in R(c)} R_{\alpha}, \quad \forall v
\]

Iteration: \(\forall c\) in the current layer \(l (l=1, 2, \ldots, M_c)\)

\[
R'_c = \Psi \left\{ \sum_{\alpha =2}^{5} \Psi \left( Q_{\alpha} - R_{\alpha} \right) \right\}
\]

\[
Q'_c = R'_c + (Q_c - R_s)
\]

Note that the \(R'_c\) term in (6) and \(Q'_c\) term in (7) are most recently updated by using values \(R_s\) and \(Q_c\) in the previous layer. An example of the algorithm with the \(4\times5\) base matrix \(H_b\) is shown in Fig. 5. This decoding scheme describes messages to be exchanged from two memory blocks, which are C2V_MEM and VN_MEM, leading to the high throughput decoder. By applying the proposed decoding architecture to
Fig. 5. Illustrative example of the block parallel LDPC decoder based on the proposed CNBPs

various structured LDPC codes, we can reduce the number of processing cycles per iteration.

Fig. 6 shows the structure of the check node-based processor using the normalized min-sum algorithm which is an approximation algorithm of the above equations (6) ~ (7) and which reduces the decoding hardware complexity. The Min-Sum module is responsible for selecting first and second minimum values and the CNBP module can apply the scaling operations, similar to the adaptive quantization method in [11]. This fully parallel architecture simultaneously reads $R_v$ messages from C2V_MEM block and $Q_v$ messages from VN_MEM block through SNs. Moreover, Read and Write operations are simultaneously performed in the dual-port memory banks. The signed magnitude and 2’s complement convert blocks are efficient for the Min-Sum module and the addition/subtraction required for calculating intermediate messages, respectively. System flexibility in terms of the supported block sizes and code rates can be achieved by the control unit without modifying CNBPs or memory blocks. The required components of CNBP, C2V_MEM, and VN_MEM are accessed through a multiplexer or fed as zero value for unused inputs.

IV. CONCLUSION

A flexible, high-throughput LDPC decoder architecture is presented for supporting different code rates and block sizes in wireless applications. The proposed CNBP architecture is suitable for block-parallel implementation and the overall decoder can achieve higher throughput than a block-serial scheduling scheme.

REFERENCES


