Design and Implementation of Multi-mode QC-LDPC Decoder

Yun Chen¹, Xiang Chen², Yifei Zhao², Chunhui Zhou², Jing Wang²
¹ School of Information Science and Technology, Xiamen University, Xiamen 361000
2 Wireless Center, RIIT, TNList, Tsinghua University, Beijing 100084
Email: {baoyapig@163.com, chenxiang98@gmail.com}

Abstract—Low-density parity-check (LDPC) codes are one of the most effective error controlling methods, which now are widely used in multiple communication systems, for example, DVB-S2, 802.16e, etc. In these communication standards, the adopted LDPC codes are all quasi-cyclic low-density parity-check (QC-LDPC) codes. The parity-check matrices of QC-LDPC codes consist of arrays of circulants, in which each row is the cyclic shift of the row above it, and the first row is the cyclic shift of the last row. This character of QC-LDPC codes results in the linear encoding complexity and opens the door for the multi-mode LDPC decoders with enough flexibility of resource multiplexing. Based on the quasi-cyclic character of QC-LDPC codes, a design method for multi-mode QC-LDPC decoders is proposed in this paper. The node memory is skillfully organized and can be expediently addressed by a simple control unit to save memory consumption in hardware implementation. Then this design method is implemented and tested on Field Programmable Gate Array (FPGA) platform. The test results show that the resource occupied by the multi-mode decoder is only a little more than that of maximal resource occupied by the single-mode decoder, and this multi-mode decoder can support at least 3 code rates with the throughput higher than 100Mbps when the iteration number is fixed to 12 and the clock frequency is set to 200MHz.

I. INTRODUCTION

Low-density parity-check (LDPC) codes, which are linear block codes defined by a very sparse parity-check matrix, were first introduced by Gallager [1]. Since LDPC codes rediscovered by Mackey and Neal in 1996 [2], more and more people are interested in these codes because of their excellent error correcting performance. With the remarkable advancement of VLSI technology, it becomes feasible to implement LDPC codes in the practical communication systems. Many decoder architectures have been proposed so far, including fully parallel implementation [3], partially parallel implementation [4] and completely serial implementation [5]. All the above-mentioned chips only support single-mode decoding. However, in order to meet different circumstances, it usually requires a decoder which can support different parity-check matrices and different code rates. So multi-mode design and reconfigurable architecture become very popular, such as [6][7][8]. In these designs, QC-LDPC codes have been adopted because of their block-structured parity-check matrices. The parity-check matrices of QC-LDPC codes consist of arrays of circulants, in which each row is the cyclic shift of the row above it, and the first row is the cyclic shift of the last row. This character of QC-LDPC codes make the memory address generation for partial-parallel decoders very efficient. Based on QC-LDPC codes, a decoder which can support 19 modes in 802.16e system is proposed [6] and a triple-mode decoder design for 802.11n system [7] is implemented. In [8], a more flexible decoder can fully support those specific 19 and 3 submatrix sizes defined in 802.16e and 802.11n. However, the above decoders only support a subset of QC-LDPC codes, such as QC-LDPC codes used in WiMAX or Wi-Fi.

In this paper, we propose a general multi-mode QC-LDPC decoder, which can support at least 3 different kinds of QC-LDPC codes. The supported codes are not limited to those codes in current standards. Instead, any QC-LDPC codes can be configured and implemented into this decoder, as they have the same expansion factor. The proposed architecture exploits the partially parallel method to guarantee the tradeoff between cost and performance. And the offset BP-based algorithm [9] has been adopted to reduce implementation complexity. Based on this multi-mode decoder structure, arbitrary configured QC-LDPC codes can be real-time decoded on this unique platform. The rest of this paper is organized as follows. Section II briefly introduces LDPC codes and QC-LDPC codes. Section III discusses the offset BP-based algorithm adopted in this design. A multi-mode decoding architecture is presented in Section IV. In Section V, the mentioned decoder is implemented in FPGA device and the results are demonstrated together with a resource occupation analysis. Finally, Section VI concludes this paper.

II. LDPC CODES DESCRIPTION

A. LDPC Codes

A LDPC code is a linear block code specified by a very sparse $M \times N$ parity-check matrix, of which few elements are ones and others are zeros. The $(n,k)$ LDPC codes mean that a $k$ bit message can be encoded to be $n$ bit codeword. Such parity-check matrix \( H \) consists of $(n-k)$ rows and $n$ columns, and the code rate is defined as $k/n$. Fig.1 (a) is an example of a $(12,6)$ LDPC code parity-check matrix. And as we all know, if $x = (x_1, \ldots, x_n)$ is a codeword, it satisfies parity-check equation:

$$H \cdot x^T = 0$$

(1)

Also, the LDPC parity check matrix \( H \) can be transformed into a graphical representation, Tanner’s graph [10], as
illustrated in Fig. 1(b). There are two sets of nodes: check nodes \((i)\) and variable nodes \((j)\). An edge between check nodes and variable nodes is drawn only if there is 1 at \((i,j)\) in the parity-check matrix. The connection represents that the mutual information can be transmitted among check nodes and variable nodes, which is the foundation of message-passing algorithm.

\[
H = \begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
\end{bmatrix}
\]

\( \text{(a)} \)

Check nodes

Variable nodes

\( \text{(b)} \)

Fig. 1. (12,6) LDPC codes (a) parity-check matrix and (b) bipartite graph

### B. QC-LDPC Codes

QC-LDPC codes, which are widely used in multiple communication systems, for example, DVB-S2, 802.16e, etc, are defined by very sparse base matrix. The element in the base matrix should be -1, 0, or a positive integer, and can be expanded to be a \(Z \times Z\) sub-matrix. The elements of -1 and 0 in base matrix denote an all-zero sub-matrix and an identity sub-matrix, respectively. An element with a positive integer (shifted value) can be expanded to be an identity matrix right-shifted by shifted value. By expanding the elements in base matrix with \(Z\), we can get a QC-LDPC parity check matrix. It means that a relatively small base matrix can be used to replace a large parity-check matrix, which makes implementation more convenient.

### III. LDPC DECODING ALGORITHM

Decoder proposed in this paper is based on the offset BP-based algorithm. The offset BP-based algorithm is a modified version of BP-Based algorithm [11], which has a tolerable performance loss compared to BP algorithm. In order to describe the algorithm easily, we define some symbols first.

- \(L(c_i)\) is the intrinsic message, \(L(r_{ji})\) is the message computed by the check node \(j\), and sent to the variable node \(i\).
- \(L(q_{ij})\) is the message computed by the variable node \(i\), and sent to the check node \(j\). Definition of \(\alpha_{ij}\), \(\beta_{ij}\) are as follows:

\[
\alpha_{ij} = \text{sign}(L(q_{ij})) \quad (2)
\]

\[
\beta_{ij} = \text{abs}(L(q_{ij})) \quad (3)
\]

Now, the offset BP-based algorithm is illustrated as follows:

- **Step 1**) Initialization
  
  Using the intrinsic messages initialize decoder.

\[
L(q_{ij}) = L(c_i) \quad (4)
\]

- **Step 2**) Check-Node Update(CNU)

Check nodes update log-likelihood ratio (LLR) in the following equation. \(\text{offset}\) is a value which can be computed by density evolution theory.

\[
L(r_{ji}) = \prod_{i \in R_{ji}} \max_{j \in R_{ij}} \left[ \frac{\alpha_{ij}}{\beta_{ij}} - \text{offset}, 0 \right] \quad (5)
\]

- **Step 3**) Variable-Node Update(VNU)

Variable nodes update LLR in the following equation.

\[
L(q_{ij}) = L(c_i) + \sum_{j \in C_i \setminus j} L(r_{ji}) \quad (6)
\]

Also calculate

\[
L(Q_j) = L(c_i) + \sum_{j \in C_i} L(r_{ji}) \quad (7)
\]

if \(L(Q_j) > 0\), the result of hard-decision is 0, otherwise the result is 1. And if all the parity check constraints are satisfied or the maximum of iterations is reached, decoding algorithm turn to Step 4), otherwise turn to Step 2).

- **Step 4**) Output

Output decoded data.

For logic circuit design with finite precision, a 6-bit quantized algorithm is adopted according to the simulation results.

### IV. DECODER HARDWARE ARCHITECTURE

Partial-parallel method [4] is very efficient for hardware implementation. The sing-mode decoder with the Partial-parallel design consists of node rams, CNU circuits and VNU circuits. And the routing between them is fixed. Based on this structure, the reconfigurable routers and a control unit are required to adapt different modes. Fig. 2 shows the block diagram of the decoder architecture.

![Block diagram of proposed decoder](image-url)
A. Input Buffers

Original LLR data are transferred to the input buffers, which consist of an array of RAMs. The number of RAMs is \( N \), which is the maximum of column of supported base matrices. The depth of each RAM is \( Z \), which is the expansion factor. So the supported maximal code length is \( N \times Z \) and the parameter \( N, Z \) can be reconfigured, respectively.

B. An Array of Node RAMs

The number of node RAMs is \( X \), which is the maximum of nonzero submatrices of supported base matrices. The results of each CNU update or VNU update are written into these node RAMs for the next update.

C. Control Unit (CU)

Control Unit (CU) is the most important part of decoder. According to value stored in mode register, it configures other modules to adapt different modes. Its chief functions are as follows:

1) Controlling input state machine to adjust input numbers.
2) Reconfiguring three routers.(input buffers and node RAM, node RAM and CNU, node RAM and VNU)
3) Selecting the number of node RAMs which need to be updated, the number of CNU circuits and the number of VNU circuits.
4) Controlling output state machine to adjust output numbers.

The key to support different modes is how to reconfigure three routers. And there are \( N \) sets of VNU circuits and \( M \) sets of CNU circuits, where \( N \) is mentioned above and \( M \) is the maximal row of supported base matrices. The mathematic function of router 1 and router 3 is mapping \( N \) into \( X \), and router 2 is mapping \( M \) into \( N \). So there is a need for 2 sets of indexes. In this paper, we analyze base matrix by Matlab and for each supported code, we generate 2 sets of indexes to achieve this mapping.

The substantial part of CU is composed of Look-up tables (LUT), as illustrated in Fig.3. According to different value in mode register, corresponding values in the table are chosen.

The chosen values reconfigure other modules to make decoder adapt different code.

![LUT of CU](image)

D. CNU Circuit

The function of CNU circuit is to compute equation (5). Firstly, it finds out the minimum, the second smallest value and mark the position of the minimum. After modifying by offset value, the output port in the position of the minimum exports the second smallest value and the output ports in the rest position export the minimum. Finally, sign bit is attached to the output data. The pipeline technique is adopted to achieve higher throughput. The number of CNU input ports is maximal check nodes degree. To support different check nodes degree, the unused input of CNU is imported as the maximum value in fixed-point representation to produce the calculated data when the number of input data is less than the hardware input ports. The block diagram of CNU circuit is shown in Fig.4.

![CNU Circuit](image)

E. VNU Circuit

The function of VNU circuit is to compute equation (6) and (7). Just like CNU, the pipeline technique is adopted. The number of VNU input ports is set to maximal variable nodes degree. And the unused pin of VNU is imported as zero value to produce the correct calculated data when the number of input data is less than the hardware input ports. As shown in Fig.4, it can be made of adders, subtractors and registers easily.

![VNU Circuit](image)

F. Output Buffers

The design of output buffers is similar to the design of interleaver. The VNU circuits output \( N \) bit hard-decision values of LLR at a time. And then the decoded information bits are written into output buffers by row. Once one frame data are stored in the output buffers, the decoded bits now are read from output buffers by column. After previous described procedure, the decoding process is accomplished.

V. FPGA IMPLEMENTATION AND RESULTS
Base on the mentioned decoder architecture in Section IV, a multi-mode decoder design which can support at most 4 different modes has been proposed. Without loss of generality, we consider the following three QC-LDPC codes to test the decoder.

<table>
<thead>
<tr>
<th>Code length</th>
<th>Code rate</th>
<th>Information Bit</th>
<th>Expansion Factor</th>
<th>Best Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>6400</td>
<td>21/25</td>
<td>5376</td>
<td>256</td>
<td>0.5</td>
</tr>
<tr>
<td>6400</td>
<td>17/25</td>
<td>4352</td>
<td>256</td>
<td>0.5</td>
</tr>
<tr>
<td>6400</td>
<td>13/25</td>
<td>3328</td>
<td>256</td>
<td>0.4</td>
</tr>
</tbody>
</table>

The optimal offset can be obtained by the computer searching and shown in column 5, Table I for different code rates. As the architecture have been described in section VI and the codes in Table I have been adopted, a multi-mode QC-LDPC decoder is implemented on Altera Stratix II EP2S90F1020I4 FPGA device. The design is described in Verilog, synthesized, placed and routed by Altera development tool quartus II 9.0.

The decoder can operate at three different code modes: 13/25 mode (default mode), 17/25 mode, 21/25 mode. By inputting “01”,“10” to mode register, the decoder can be changed into 17/25 mode or 21/25 mode, respectively. To understand the additional overhead caused by the reconfiguration, Table II shows the resource utilization statistics of the multi-mode decoder and the single-mode decoder (code rate 13/25), which is the maximal resource consumption of the single-mode decoder.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ALUTs</td>
<td>9180(13%)</td>
<td>12662(17%)</td>
</tr>
<tr>
<td>Registers</td>
<td>9934(14%)</td>
<td>12088(17%)</td>
</tr>
<tr>
<td>Block Memory Bits</td>
<td>276819(6%)</td>
<td>311171(7%)</td>
</tr>
<tr>
<td>Block ram</td>
<td>129(14%)</td>
<td>180(20%)</td>
</tr>
</tbody>
</table>

As we can see in Table II, the resource occupied by the multi-mode decoder is only a little more than that of maximal resource occupied by the single-mode decoder.

In summary, the parameters of this multi-mode decoder are shown in Table III. The results show that this decoder is very flexible for many parameters can be reconfigured and can support at least 3 code rates with the throughput higher than 100Mbps when the iteration number is fixed to 12 and the clock frequency is set to 200MHz.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>The proposed decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Implement 3 At most 4</td>
</tr>
<tr>
<td>Codeword</td>
<td>6400(reconfigurable)</td>
</tr>
<tr>
<td>Code rate</td>
<td>13/25,17/25,21/25(reconfigurable)</td>
</tr>
<tr>
<td>Expansion Factor</td>
<td>256(reconfigurable)</td>
</tr>
<tr>
<td>Iteration Number</td>
<td>12(reconfigurable)</td>
</tr>
<tr>
<td>Supported Check Nodes Degree</td>
<td>4–32</td>
</tr>
<tr>
<td>Supported Variable Nodes Degree</td>
<td>3–13</td>
</tr>
<tr>
<td>Frequency(MHz)</td>
<td>200</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

In this paper, a multi-mode QC-LDPC decoder design method and relative architecture are proposed. This multi-mode QC-LDPC decoder is base on a partially parallel architecture and the offset BP-based algorithm is employed. Any type of QC-LDPC codes with the same expansion factor can be supported and at least 3 different QC-LDPC codes can be configured into this unique architecture. FPGA implementation and tests show that this multi-mode QC-LDPC decoder can achieve more than 100Mbps throughput at a clock frequency 200MHz for each code and the resource consumption is only a little more than the maximal one of the single-mode decoder.

REFERENCES