This paper presents a method for constructing a pipeline-aware quasi-cyclic irregular repeat accumulate low-density parity-check (PA-QC-IRA-LDPC) codes and efficient rate-1/2 (2016, 1008) PA-QC-IRA-LDPC decoder architecture. A novel pipeline scheduling method is proposed. The proposed methods efficiently reduce the critical path using pipeline without any bit error rate (BER) degradation. The proposed pipeline-aware LDPC decoder provides a significant improvement in terms of throughput, hardware efficiency, and energy efficiency. Synthesis and layout of the proposed architecture is performed using 90-nm CMOS standard cell technology. The proposed architecture shows more than 53% improvement of area efficiency and much better energy efficiency compared to the previously reported architectures.

**Abstract**

This paper presents a method for constructing a pipeline-aware quasi-cyclic irregular repeat accumulate low-density parity-check (PA-QC-IRA-LDPC) codes and efficient rate-1/2 (2016, 1008) PA-QC-IRA-LDPC decoder architecture. A novel pipeline scheduling method is proposed. The proposed methods efficiently reduce the critical path using pipeline without any bit error rate (BER) degradation. The proposed pipeline-aware LDPC decoder provides a significant improvement in terms of throughput, hardware efficiency, and energy efficiency. Synthesis and layout of the proposed architecture is performed using 90-nm CMOS standard cell technology. The proposed architecture shows more than 53% improvement of area efficiency and much better energy efficiency compared to the previously reported architectures.

**Keywords**: Low-density-parity-check (LDPC), min-sum decoder, pipeline-aware, forward error correction (FEC).

---

I. Introduction

Low-density parity-check (LDPC) codes are widely used in modern communication systems. LDPC codes are one of the most important forward error correction (FEC) codes because of their remarkable performance that can approach Shannon limit. Recently, several techniques are proposed that
offer wide variety of tradeoffs between code performance (BER), decoder complexity and decoder throughput. These techniques can be classified into three major categories. 1) Construction of LDPC code for low complexity decoder architecture with minimal bite error rate (BER) degradation\textsuperscript{[1, 2]}. These codes are known as "block structured LDPC codes" or "architecture-aware LDPC codes". 2) Improvement in decoding algorithm/schedule to achieve faster convergence and low complexity decoding. Min-Sum algorithm\textsuperscript{[3]} offers low complexity compared to log belief propagation (log-BP) algorithm. Layered schedule\textsuperscript{[4]} offers faster convergence compared to the conventional BP schedule. Variable node units (VNUs) are updated more often compared to BP. Hence, faster convergence is achieved and the similar BER is achieved with almost 50% of the iterations. 3) Efficient hardware design techniques such as non-uniform quantization\textsuperscript{[5]}, differential shifting\textsuperscript{[6]}, pipelining\textsuperscript{[7]}, the partially parallel decoder designs\textsuperscript{[8]}, multi-mode QC-LDPC decoder architecture\textsuperscript{[9]}. The focus of this work is on the first and third category. A code construction method of pipeline-aware Quasi-Cyclic Irregular Repeat Accumulate (QC-IRA) LPDC code is described. An efficient decoder design is proposed to achieve low-latency and high-throughput without any BER degradation.

The rest of this paper is organized as follows. Section II describes a brief introduction of QC-IRA LDPC codes and decoding algorithm. Section III described the code construction and efficient decoder architecture for pipeline-aware QC-IRA-LDPC code. Performance comparison and results are presented in Section IV. Finally, conclusions are drawn in Section V.

II. QC-IRA-LDPC CODE AND DECODING ALGORITHM

1. IRA-LDPC Code

Repeat accumulate (RA) or irregular repeat accumulate (IRA) codes can be considered to be a concatenation of a set of repetition codes with an accumulator, through an interleaver\textsuperscript{[8-9]}. Despite their simple structure, they were shown to provide good performance\textsuperscript{[9]} and, more importantly, they paved a path toward the design of efficiently encode-able and decode-able LDPC codes.

An RA code consists of a serial concatenation of a single rate-1/q repetition code through an interleaver with an accumulator having transfer function 1/(1+qD)\textsuperscript{[8-9]}. RA codes can be either non-systematic or systematic. In the first case, the accumulator output, p, is the codeword and the code rate is 1/q. For systematic RA codes, the information word, u, is combined with p to yield the codeword c = [u p] and so that the code rate is 1/(1+q). The systematic irregular repeat accumulate (IRA) codes generalize the systematic RA codes and the repetition rate may differ across the k information bits. Irregular repeat–accumulate codes provide several advantages over RA codes. They allow both flexibility in the choice of the repetition rate for each information bit so that high rate codes may be designed and capacity is more easily approached. The parity check matrix with k information bits and m parity bits (m = n - k) for systematic RA and IRA codes has a following form.

\[
H = \begin{bmatrix} H_0 & H_1 \end{bmatrix}
\]

where \(H_0\) is an m\times m "dual-diagonal" square matrix.

2. QC-IRA-LDPC Code

Given the code rate, length, and degree distributions, an IRA code is defined entirely by the matrix \(H_0\). While a random-like \(H_0\) would generally give good performance, it is problematic for both encoder and decoder implementations. An IRA-LDPC code with structured (quasi cyclic) \(H_0\) developed in\textsuperscript{[8-9]} actually showed a good performance compared to random like codes. QC-IRA-LDPC codes can be developed by choosing \(H_0\) to be array of circulant
permutation matrices. Let \( P \) be an \( L \times J \) array of \( Q \times Q \)
circulant permutation matrices then \( HQC \) is defined as
(2).
\[
HQC = [ P \ II H_{p}I_{T}] \tag{2}
\]
\[
P = \begin{bmatrix}
  b_{0,0} & b_{0,1} & \cdots & b_{0,J-1} \\
  b_{1,0} & b_{1,1} & \cdots & b_{1,J-1} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{L-1,0} & b_{L-1,1} & \cdots & b_{L-1, J-1}
\end{bmatrix} \tag{3}
\]
\[
\prod H_{p} \Pi^{T} = \begin{bmatrix}
  I_{0} & I_{0} & \cdots & I_{0} \\
  I_{0} & I_{0} & \cdots & I_{0} \\
  \vdots & \vdots & \ddots & \vdots \\
  I_{0} & I_{0} & \cdots & I_{0}
\end{bmatrix}
\tag{4}
\]
where, parity part of \( HQC \), i.e., \( \Pi \Pi^{T} \) is a
quasi-cyclic dual diagonal matrix, with upper right
entry may be included for tail-biting accumulator. \( I_{0} \) is a
\( Q \times Q \) identity matrix and \( I_{l} \) is a cyclic shifted
version of \( I_{0} \). Generally, inverse of the both parts of
\( HQC \) is required to obtain generator matrix. The
inverse only exists only when tail-biting is absent.

3. Layered Decoding Algorithm
A Min–Sum based layered decoding algorithm is briefly
described below.
\[
L'[k] = P^{l-1}[k] - R'[k-1] \tag{5}
\]
\[
R'[k] = a \times \prod_{\text{all}(c \rightarrow \nu)} \text{sign}(L'[k]) \times \text{Min}_{\text{all}(c \rightarrow \nu)} |L'[k]| \tag{6}
\]
\[
P'[k] = L'[k] + R'[k] \tag{7}
\]
Here, \( l \) denotes the layer and \( k \) is the iteration
number. \( R \) denotes the check to variable message \( c \)
to \( \nu \). \( L \) represents the variable to check message \( \nu \)
to \( c \). In the \( k \)-th iteration, the LLR message from
layer \( l \) to next layer for variable node \( \nu \) is
represented by \( P \). In (6), \( N(c) \) / \( \nu \) denotes the set of
variable nodes connected to check node \( c \) excluding
variable node \( \nu \). The layered decoder performs all
operations from (5) to (7) in a sequence.

III. PIPELINE–AWARE QC–IRA–LDPC CODE
CONSTRUCTION

1. Pipelining Issues for LDPC Decoders
Efficient pipelining is an essential part of high
throughput LDPC decoders. But in case of layered
decoder, pipelining is not straightforward. A pipeline
method with approximated layered decoding approach
causes BER degradation. Hence this technique is
not much efficient in terms of BER performance. On
the other hand, [10] used a multi-frame pipeline
technique, which effectively doubles the memory
requirement.

The main issue is the data dependency between
consecutive rows in the layered decoding algorithm,
because a posterior probability messages from layer
"j-1" are required before starting the process for
layer "j". Quasi-cyclic \( H \) matrix with four layers or
block rows and a one stage pipeline schedule is
shown in the Fig. 1. Block layer \# 0 and 1 can be
processed independently, because these layers have
no conflicting entry. Similarly block rows \# 3 and 4
have no conflicting entry. But still pipeline insertion
is not possible, because column five and seven have
a conflicting entry. If these two conflicting entries

\[
L0 A L1 A L2 A L3 A L0 A L1 A L2 A L3 A
\]
\[
L0 B L1 B L2 B L3 B L0 B L1 B L2 B L3 B
\]

Fig. 1. Pipeline conflict in entries of H-matrix for LDPC
decoders.
are removed by any mean, then it is possible to insert a pipeline that follows a schedule shown in Fig. 1. Hence, any LDPC code that follows the following lemma (with \( i \geq 2 \)) is considered as a pipeline-aware LDPC code.

*If all \( i \) consecutive block rows (layers) have no 1-entry in common then, \( i-1 \) pipeline stages can be inserted without any pipeline hazard.*

2. Construction of Pipeline-Aware QC-IRA-LDPC Codes

An example of dual-diagonal 8×8 quasi-cyclic matrix is shown in Fig. 2(a). It is quite clear that it doesn’t depict a pipeline-aware property. A row permutation is performed on this matrix. First all the even rows i.e. rows \( \# 0, 2, 4, 6 \) are permuted to row \( \# 0, 1, 2, 3 \). Then all odd rows i.e. rows \( \# 1, 3, 5, 7 \) are permuted to row \( \# 4, 5, 6, 7 \). The resulting matrix shown in Fig. 2(b) is a pipeline-aware with “\( i > 2 \)”.

The QC-IRA-LDPC code construction method described in [8, 9] has been improved and tailored to achieve the pipeline-aware property described above. Let \( H_Q \) is defined by (2) then, systematic and parity parts are given in (3) and (4), respectively. A modified construction method for QC-IRA-LDPC code with “\( i = 2 \)” pipeline-aware feature is described as follows.

\[
\begin{bmatrix}
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
\end{array}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
\end{array}
\end{bmatrix}
\]

(a) (b)

*Fig. 2.* (a) 기존의 dual-diagonal matrix
(b) 제안한 permuted pipeline-aware matrix

1. Generate \( L \times J \) quasi cyclic matrix \( H_0 \):
   - a) Initialize \( H_0 \) as quasi cyclic dual diagonal matrix.
   - b) Permute \( H_0 \) according to the odd/ even splitting method described above.

2. Generate \( L \times J \) quasi cyclic matrix \( P \):
   - a) Randomly select \( j^{th} \) column and \( i^{th} \) row (Do not select a \([i, j]\) block which already has a cyclic permutation registered).
   - b) Check pipeline-aware condition shown in (8) and (9):
     \[
     \begin{align}
     b_{0-i,j} &= 0 \quad (8) \\
     b_{9-0,j} &= 0 \quad (9)
     \end{align}
     \]
   - Here, \( 0_0 = Q \times Q \) zero matrix or empty matrix.
   - c) If condition ‘2b’ is true then go to step \# 3, else go to step \# 2.

3. Assign a cyclic permutation value to \( b_{i,j} \):
   - a) Randomly select an integer \( x \in [0, Q) \), different from most recently selected integer and set \( b_{i,j} = x \).

4. Check condition 1 and 2 described below [8]
   - a) Condition 1: Every entry in each column of \( P \) should have a distinct cyclic permutation value.
   - b) Condition 2: Check all 4-cycles with in \( P \) by applying (10) and check all length-4 cycles across \( P \) by applying (11).

\[
\begin{align}
\left( b_{l_1,j} - b_{l_2,j} \right) \mod Q \neq \left( b_{l_2,j} - b_{l_2,j} \right) \mod Q \quad (10) \\
\left( b_{l_1,j} - b_{l_2,j} \right) \mod Q \neq 1 \quad (11)
\end{align}
\]

where,

\[
l_1, l_2 \in [0, L), l_1 \neq l_2 \land j_1, j_2 \in [0, J), j_1 \neq j_2
\]

and

\[
l \in [0, L-1) \land j \in [0, J)
\]
c) If both conditions ‘4a’ or ‘4b’ are true, then register $b_{1j}$ at $j^{th}$ column and $r^{th}$ row and go to step # 2, else go to step # 3.

The algorithm described above avoids all 4-cycles. Hence the girth of the code is at least 6. Further improvement is possible by applying the 6-cycle condition given in [9].

3. Pipelined-Aware QC-IRA-LDPC Decoder Architecture

The proposed pipeline-aware QC-IRA-LDPC decoder architecture with row weight of seven is shown in Fig. 3(a). First the switch network rotates all the input values and old check to variable values are retrieved from memory simultaneously. Then subtraction is performed according to Eq. (5). Then CNU operation is carried out. CNU performs the minimum sorting and sign operation shown in (6). Efficient CNU architecture is adopted from [11]. First stage of CNU is compared and selected, and then two CU stages are added. The pipeline stage can be freely adjusted inside CNU without any pipeline hazard because the proposed code is pipeline-aware. We have inserted the pipeline just before the CU stages to reduce the critical path. The output of CNU (first and second minimum values) or new check to variable node messages are scaled with factor $\alpha=0.75$. Finally, addition operation is performed using (7), to generate new log-likelihood ratio (LLR) values, for next layer. The process continues until last layer in final iteration is reached. Fig. 3(b) shows the pipeline schedule for 12 layers. Each block in pipeline schedule is represented by a layer number and a pipeline section. The pipeline section is shown as “Section A” and “Section B” in Fig. 3(a). It is clear form pipeline schedule that two different layers are processed by “Section A” and “Section B” simultaneously. There is no pipeline hazard because pipeline-aware property made all successive layers conflict free.

![Diagram](image)

**Fig. 3.** (a) Proposed pipeline-aware LDPC decoder architecture. (b) Pipeline schedule for the proposed LDPC decoder.

IV. RESULTS AND COMPARISONS

1. BER Performance

The proposed method was used to design rate-1/2 (2016, 1008) PA-QC-IRA-LDPC code with 12-layers or block rows, pipeline “$r=2$” and “$Q=84$”. Fig. 4 shows the floating-point BER performance of rate-1/2 (2016, 1008) QC-IRA-LDPC code without pipeline condition and proposed pipeline-aware QC-IRA LDPC code with pipeline condition (using 10
The proposed pipeline-aware layered decoder architecture with various other decoders of comparable code length \cite{12-15}. The proposed code length is 107-bit. The nearest code lengths available in literature are 1944-bit \cite{14}, 2304-bit \cite{12, 14}, and 3072-bit \cite{15} LDPC decoders given in \cite{12-15} implement various techniques for efficient decoding. To provide a fair comparison, the factors of area and energy efficiency are introduced. Area efficiency is measured in throughput provided for each mm$^2$ of area. The proposed decoder shows much better area efficiency compared to all other works provides in \cite{12-15}. The factor of power efficiency shows how much joules of energy is consumed for each bit in each iteration. The factor of $pJ$/bit/iteration is much better for comparison with various code lengths because this factor not only encompasses power but also the time required to produce a decoded output.

The proposed decoder developed with pipeline-aware technique shows a major advantage in terms of throughput with reduced area. The K Zhang decoder \cite{12} shows an impressive throughput of 2.2 Gb/s with 2.9 mm$^2$ of area. Whereas the proposed decoder shows about 40% throughput improvement.
compared to [12]. It outperforms the decoder given in [12] by more than 53% in terms of area efficiency. The proposed decoder shows much better power efficiency primarily because it consumes less power and it requires much less time to give a decoded output. The proposed decoder only needs \((\text{layers} \times \text{Iter}) + 121\) clock cycles, whereas the K. Zhang decoder needs 990 clock cycles\(^{[13]}\). Hence it consumes much more pico joules over a period of time.

The C. Roth decoder\(^{[14]}\) achieves less than a 1.0 Gb/s with the area lesser than the proposed decoder. But in terms of area efficiency the proposed decoder shows much better results. In terms of power efficiency both the decoders depict almost similar results, but the proposed decoder shows improved power without considering register transfer level optimizations utilized in [14]. The reduced power in the proposed decoder is due to efficient pipeline-aware technique discussed in section III.

The Y. Sun decoder [13] shows 1 Gb/s of throughput with 3.5 mm\(^2\) of area. The proposed decoder shows a better area efficiency compared to [13]. The power efficiency depends upon the power and time required to get the decoded output. We cannot find enough information in [13] to accurately calculate the power efficiency.

The 2048-bit decoder shown in [15] is synthesized using 180-nm CMOS technology. Hence some conversion method is required to provide fair comparison between decoder [15] and the proposed decoder. We have adopted the conversion factors form in [16]. A factor of \((1.4142)^2 = 4\) is used for area conversion and a factor of \((1.4142)^4 = 2\) is used for throughput conversion. A factor of \((1.4142)^2 = 4\) is used to convert power efficiency. The proposed decoder shows much better area and energy efficiency compared to [15].

V. CONCLUSION

The paper introduces a novel construction method for PA–QC-IRA-LDPC code. A pipeline stage can be inserted inside the decoder architecture without BER degradation. Moreover, the proposed decoder offers an acceptable increase in throughput with reduced area, compared to the other decoders with comparable code length. The proposed decoder shows much better area and energy efficiency compared to other decoders with similar code lengths available in literature. The proposed techniques are expected to be incorporated in next-generation high efficiency QC-LDPC decoder design.

REFERENCES


저자 소개

Saboo Ajaz(학생회원)
2006년 NED University of Engg. &Tech. Karachi, 전자공학과 학사 졸업
2010년 University of Wollongong, Australia, Electrical Computer and Telecom Engineering 석사 졸업
2011년 ~ 현재 인하대학교 정보통신공학과 박사 재학

<주관심분야 : 오류정정 아키텍처 설계>

이한호(평생회원)-교신저자
1993년 충북대학교 전자공학과 학사 졸업
1996년 Univ. of Minnesota 전기 컴퓨터공학 석사 졸업
2000년 Univ. of Minnesota 전기 컴퓨터공학 박사 졸업
2000년~2002년 Member of Technical Staff, Lucent Technologies (Bell Labs.), USA.
2002년~2004년 Assistant Prof. Dept. of Electrical and Computer Engineering, Univ. of Connecticut, USA.
2004년~현재 인하대학교 정보통신공학과 교수
<주관심분야 : 디지털신호처리 및 오류정정 아키텍처 설계>