Finite Alphabet Iterative Decoders for LDPC codes: Optimization, Architecture and Analysis

Fang Cai, *Student Member, IEEE*, Xinmiao Zhang, *Senior Member, IEEE*, David Declercq, *Senior Member, IEEE*, Shiva Kumar Planjery, *Student Member, IEEE*, Bane Vasic, *Fellow, IEEE*

Abstract—Low-density parity-check (LDPC) codes are adopted in many applications due to their Shannon-limit approaching error-correcting performance. Nevertheless, belief-propagation (BP) based decoding of these codes suffers from the error-floor problem, *i.e.*, an abrupt change in the slope of the error-rate curve that occurs at very low error rates. Recently, a new type of decoders termed finite alphabet iterative decoders (FAIDs) were introduced. The FAIDs use simple Boolean maps for variable node processing, and can surpass the BP-based decoders in the error floor region with very short word length. We restrict the scope of this paper to regular $d_v = 3$ LDPC codes on the BSC channel. This paper develops a low-complexity implementation architecture for the FAIDs by making use of their properties. Particularly, an innovative bit-serial check node unit is designed for the FAIDs, and a small-area variable node unit is proposed by exploiting the symmetry in the Boolean maps. Moreover, an optimized data scheduling scheme is proposed to increase the hardware utilization efficiency. From synthesis results, the proposed FAID implementation needs only 52% area to reach the same throughput as one of the most efficient standard Min-Sum decoders for an example (7807, 7177) LDPC code, while achieving better error-correcting performance in the error-floor region. Compared to an offset Min-Sum decoder with longer word length, the proposed design can achieve higher throughput with 45% area, and still leads to possible performance improvement in the error-floor region.

I. INTRODUCTION

Low-density parity-check (LDPC) codes are used in many applications due to their excellent error-correcting performance. Traditionally, LDPC codes are decoded by the iterative belief propagation (BP) algorithm [1] or its approximations, such as the Min-Sum algorithm [2], with small performance loss. It is now well-known that LDPC codes can asymptotically approach the Shannon limit. However, on finite-length codes, the error-correcting performance curve in the low error-rate region can flatten out due to the presence of cycles in the corresponding Tanner graph. This is called the error floor. It happens because the decoding can converge to trapping sets [3], [4] and fail to correct all errors even if more decoding iterations are carried out. The performance in the

error-floor region is of critical importance for applications that require very low error rates, such as flash memory and optical communications. Intensive research has been conducted to lower the error floor [5]–[7]. However, existing approaches need either multiple decoding trials with high-overhead post-processing or complicated variable node processing.

Recently, a new type of decoders referred to as finite alphabet iterative decoders (FAIDs), were introduced for LDPC codes [8], [12]. In these decoders, the messages are represented by alphabets with a very small number of levels, and the variable-to-check (v-to-c) messages are derived from the check-to-variable (c-to-v) messages and channel information through a predefined Boolean map that is designed to optimize the error-correcting capability in the error floor region. It has been shown that with only an alphabet size of seven levels in the message quantization, which translates to messages of 3-bit word length, the FAIDs can outperform floating-point BP decoders in the error-floor region over the binary symmetric channel (BSC). In addition, multiple FAIDs with different map functions can be adopted to further improve the performance at the cost of higher complexity [13]. Although the current stage of development of FAIDs is restricted to column-weight-three LDPC codes and the BSC channel, this particular case is important for magnetic recording applications, like *e.g.* hard-drives or flash-memories.

In this paper, more detailed analysis and simulation results are provided for FAIDs, and a low-complexity implementation architecture is proposed for FAIDs by making use of their properties. We show that FAIDs with seven-level alphabets can also outperform the offset Min-Sum decoder with 6-bit word length in the error-floor region over the BSC. Since the message word length is very short in FAIDs, adopting bit-serial check node units (CNUs) and processing all the v-to-c messages connected to a check node simultaneously leads to higher efficiency in the decoder. An innovative bit-serial CNU architecture is developed for FAIDs based on the architecture in [10], which introduced one of the most efficient designs for Min-Sum decoders. As opposed to the design in [10], both 'Min1' and 'Min2' are computed in our CNU to prevent performance loss on the FAIDs. Moreover, the Boolean maps at the variable node units (VNUs) of FAIDs are symmetric and lead to small area requirement. An optimized data scheduling scheme is also proposed in this paper to maximize the hardware utilization efficiency. For a (7807, 7177) quasi-cyclic (QC) LDPC code, synthesis results show that the proposed FAID implementation with 7-level alphabets only requires 52% area to reach the same throughput

This material is based upon work supported by the National Science Foundation under Grants ECCS-0846331, ECCS-0802159, CCF-0963726, CCF-1314147 and Institut Universitaire de France.

Fang Cai and X. Zhang are with the Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106 (e-mail: {xinmiao.zhang, fang.cai}@case.edu). D. Declercq and S. K. Planjery are with ETIS Lab, ENSEA-Université de Cergy-Pontoise-CNRS, UMR 8051, 95014 Cergy-Pontoise, France (email: {declercq, shiva.planjery}@ensea.fr). B. Vasic is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721 (email: vasic@ece.arizona.edu).

as the standard Min-Sum decoder in [10] and 45% area to reach higher throughput than the offset Min-Sum decoder with longer word length, while achieving better performance in the error-floor region. Parts of this paper have appeared in [11].

The outline of this paper is as follows. Section III introduces the framework of FAIDs highlighting their differences compared to Min-Sum-based decoders and provides simulation results. We further demonstrate that the use of several FAIDs update functions in the decoder leads to a substantial performance gain in the error floor, and targets guaranteed error correction of the LDPC decoder despite the presence of unavoidable trapping sets in the Tanner graph of the code. The proposed decoder architectures are presented in Section IV. Section V analyzes the hardware complexity and provides comparisons with Min-Sum decoders. Conclusions are drawn in Section VI.

II. PRELIMINARIES

An (N, K) binary LDPC code is a linear block code that can be defined by its corresponding parity check matrix H or the associated Tanner graph G . The Tanner graph G is a bipartite graph that consists of a set of variable nodes $V = \{v_1, \dots, v_N\}$ and a set of check nodes $C = \{c_1, \dots, c_M\}$. In the graph G , a check (variable) node represents a row (column) of H , and a check node is connected to a variable node if the corresponding entry in H is nonzero. The H matrix of a QC-LDPC code consists of $r \times t$ sub-matrices of dimension $L \times L$. Each sub-matrix can be either a cyclically shifted identity matrix or a zero matrix. Due to the regularity in H , QC-LDPC codes usually have more efficient partial-parallel hardware implementations.

LDPC codes are decoded using message-passing decoders, where messages are passed iteratively along the edges of the graph G between check and variable nodes until a valid codeword is found or the maximum number of iterations is reached. A v-to-c message is determined by the VNU function using the channel information and the c-to-v messages from all of its other neighboring check nodes that were computed in the previous iteration. Similarly, the c-to-v message to a variable node is calculated by the CNU function based on the v-to-c messages from all other neighboring variable nodes. Note that an important feature of message-passing decoders is that during the calculation of an outgoing message on a particular edge of the node, all incoming messages except the one on the particular edge are used in the calculation. Such incoming messages shall be referred to as *extrinsic incoming* messages.

Let $\mathbf{y} = (y_1, y_2, \dots, y_N)$ be the input to a message-passing decoder where each y_i , also referred to as *channel value*, is calculated based on the information received by the variable node v_i from the channel. In this paper, we restrict our focus to column-weight-three codes and the BSC.

III. FINITE ALPHABET ITERATIVE DECODERS

A. Definitions

We now describe the general framework of FAIDs that was introduced in [12] for LDPC codes. An N_s -level FAID

denoted by D is defined as a 4-tuple given by $D = (\mathcal{M}, \mathcal{Y}, \Phi_v, \Phi_c)$. The finite alphabet $\mathcal{M}$ defined as $\mathcal{M} = \{-L_s, \dots, -L_1, 0, L_1, \dots, L_s\}$, where $L_i \in \mathbb{R}^+$ and $L_i > L_j$ for any $i > j$, consists of $N_s = 2s + 1$ levels for which the message values are confined to. The sign of a message $x \in \mathcal{M}$ can be interpreted as the estimate of the bit associated with the variable node for which x is being passed to or from (positive for zero and negative for one), and the magnitude $|x|$ as a measure of how reliable this value is.

The set $\mathcal{Y}$, which denotes the set of possible channel values, is defined for the case of BSC as $\mathcal{Y} = \{\pm C\}$.

For the n -th symbol of the codeword, the channel value $y_n \in \mathcal{Y}$ corresponding to node v_n is determined based on its received value. The mapping $0 \rightarrow C$ and $1 \rightarrow -C$ is used in this paper. Let $m_1, \dots, m_{l-1}$ denote the extrinsic incoming messages to a node with degree l .

The CNU function $\Phi_c : \mathcal{M}^{d_c-1} \rightarrow \mathcal{M}$ used for the update at a check node with degree d_c is given by

$$\Phi_c(m_1, \dots, m_{d_c-1}) = \left(\prod_{j=1}^{d_c-1} \text{sgn}(m_j) \right) \min_{j \in \{1, \dots, d_c-1\}} (|m_j|), \quad (1)$$

where sgn denotes the sign function.

The VNU function $\Phi_v : \mathcal{Y} \times \mathcal{M}^{d_v-1} \rightarrow \mathcal{M}$ used for the update at a variable node v_n , $n = 0 \dots N-1$ with degree d_v , can be described as a closed-form function given by

$$\Phi_v(m_1, m_2, \dots, m_{d_v-1}, y_i) = Q \left(\sum_{j=1}^{d_v-1} m_j + \omega_n \cdot y_n \right), \quad (2)$$

where the function $Q(\cdot)$ is defined as follows based on a threshold set $\mathcal{T} = \{T_i : 1 \leq i \leq s+1\}$ such that $T_i \in \mathbb{R}^+$ and $T_i > T_j$ if $i > j$, and $T_{s+1} = \infty$.

$$Q(x) = \begin{cases} \text{sgn}(x)L_i, & \text{if } T_i \leq |x| < T_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

The weight ω_i assigned to the channel value in Eq. (2), which is one of the main difference of FAIDs compared to the state-of-the-art decoders. It is computed from a symmetric function $\Omega : \mathcal{M}^{d_v-1} \rightarrow \mathbb{R}^{\geq 0}$ whose input arguments are the $d_v - 1$ incoming messages of a VNU. The function Ω could be linear or non-linear, and its purpose is to modify the output of the VNU update in order to prevent the failure of the message passing decoder on specific small topologies of error events referred to as Trapping sets. The careful design of Ω through a systematic analysis of the dominant Trapping sets of regular $d_v = 3$ LDPC codes is the key feature of the FAID framework (see section III-C). As a result, the Ω function, and the corresponding FAID, are designed to improve the error-rate performance in the error floor region.

Note that a particular FAID is uniquely specified by the choice of the map for Φ_v as the function Φ_c is the same in all FAIDs considered. Furthermore, the function Φ_v must satisfy the following two properties.

Property 1 (Property of symmetry):

$$\Phi_v(y_n, m_1, \dots, m_{d_v-1}) = -\Phi_v(-y_n, -m_1, \dots, -m_{d_v-1})$$

Property 2 (Property of monotonicity):

$$\Phi_v(y_n, m_1, \dots, m_{d_v-1}) \geq \Phi_v(y_n, m'_1, \dots, m'_{d_v-1}) \text{ when } m_j \geq m'_j \forall j \in \{1, \dots, d_v - 1\}.$$

Alternatively, for column-weight-three codes, the VNU function Φ_v can be represented as a simple two-dimensional Boolean map or look-up table (LUT) that is defined for a specific channel value. Table I shows an example of a Boolean map defining Φ_v of a 7-level FAID when the channel value is -C. The corresponding map for +C can be deduced from symmetry.

TABLE I
BOOLEAN MAP DEFINING THE VNU OF A 7-LEVEL FAID WHEN $y_n = -C$

m_1/m_2	-L3	-L2	-L1	0	+L1	+L2	+L3
-L3	-L3	-L3	-L3	-L3	-L3	-L3	0
-L2	-L3	-L3	-L3	-L2	-L2	-L1	L1
-L1	-L3	-L3	-L2	-L2	-L1	0	L2
0	-L3	-L2	-L2	-L1	0	L1	L2
L1	-L3	-L2	-L1	0	0	L1	L2
L2	-L3	-L1	0	L1	L1	L2	L3
L3	0	L1	L2	L2	L2	L3	L3

At the end of each decoding iteration, the hard-decision bit corresponding to each variable node v_i is determined as the sign of $y_i + \sum_{j=1}^{d_v} m_j$.

B. Min-Sum-Based Decoders: Instances of FAIDs

For comparisons with FAIDs in this paper, we consider two quantized Min-Sum-based decoders: the standard Min-Sum decoder, and the offset Min-Sum decoder. The VNU function Φ'_v used in both the decoders is given by

$$\Phi'_v(y_n, m_1, \dots, m_{d_v-1}) = y_n + \sum_{j=1}^{d_v-1} m_j. \quad (3)$$

where the value y_n can take 2 values for the BSC channel $\{-C, +C\}$.

The CNU for the standard Min-Sum decoder is the same as the function used in FAIDs, which is given by (1). From (1), it is clear that the c-to-v messages from a check node can only have two possible magnitudes, which are the minimum and the second minimum among the magnitudes of all incoming v-to-c messages. Let the two magnitudes be denoted by Min1 and Min2 respectively. As a result, only four values need to be recorded for each check node: Min1, Min2, $S = \prod_{j=1}^{d_c} \text{sgn}(m_j)$, and the index of the variable node, I , that provides Min1. Then the message to the variable node with index I has magnitude Min2 and the message to all other variable nodes have Min1. The sign of each c-to-v message can be computed as multiplying S with the sign of the corresponding v-to-c message.

The CNU for the offset Min-Sum decoder [2] involves the

introduction of an offset factor γ , and is given by

$$\Phi'_c(m_1, \dots, m_{d_c-1}) = \left(\prod_{j=1}^{d_c-1} \text{sgn}(m_j) \right) \max \left(\min_{j \in \{1, \dots, d_c-1\}} (|m_j|) - \gamma, 0 \right) \quad (4)$$

It is evident from (4) that in addition to the calculation of the four values Min1, Min2, S , and the index I , the offset factor γ must be subtracted from each magnitude in order to determine the outgoing message. The offset factor γ serves to reduce the overestimate of the outgoing message produced by a check node using (1), especially when the magnitudes of the incoming messages are small. Therefore, the offset factor is also often referred to as a *correction factor*. An appropriate choice of the offset factor enables the decoder to achieve a performance approaching the performance of BP in the waterfall region while also offering possible improvement in the error floor region.

Note that in the representation of the offset Min-Sum decoder, while the offset factor is introduced at the CNU as shown in (4), an equivalent representation can be obtained by introducing the offset factor at the VNU instead. Using this equivalent representation, it can be shown that both the standard Min-Sum and the offset Min-Sum decoders are instances of FAIDs. This is due to the property of symmetry and property of monotonicity that the VNU function Φ_v must satisfy. As an example, Table II shows the VNU function for a 3-bit offset Min-Sum decoder as a Boolean map which can also be treated as an instance of a 7-level FAID.

As can be seen in Table II, the offset corrected Min-Sum has a very regular organisation in its LUT representation, and the only non-linearity can be seen in the sequence of three zeros in each column of the LUT, which is due to the offset correction. This non-linearity is already of great help in the iterative decoding process since the offset corrected Min-Sum provide much greater error correction capability than the simple Min-Sum. The LUT of the FAID decoder in Table I shows even more non-linearities than the offset corrected Min-Sum. One example is the fact that the amplitude gaps between two adjacent squares could be greater than one, with the extreme case that $l_{1,6} = -L_3$ and $l_{1,7} = 0$. Although it is difficult to directly relate those non-linearities to a particular effect in terms of improved error correction, the FAID analysis that we provide in [12], [13], and in this paper, shows that a specific organisation of these non-linearities in the LUT can lead to better iterative decoders than the classical ones, especially in the error floor region.

C. Discussion on the Design of FAIDs

The methodology used for designing FAIDs relies on the knowledge of potentially harmful structures called *trapping sets* [3] that could be present in the Tanner graph of the code, and which cause conventional iterative decoders to fail for certain low-weight error patterns. A notation typically used to denote a trapping set (TS) is (a, b) where a is the number of variable nodes and b is the number of odd-degree check nodes in the subgraph induced by the variable nodes [3].

TABLE II
VNU OF A 3-BIT OFFSET MIN-SUM REPRESENTED AS A FAID

m_1/m_2	$-L_3$	$-L_2$	$-L_1$	0	$+L_1$	$+L_2$	$+L_3$
$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$
$-L_2$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	0
$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	0	0
0	$-L_3$	$-L_3$	$-L_2$	$-L_1$	0	0	0
L_1	$-L_3$	$-L_2$	$-L_1$	0	0	0	L_1
L_2	$-L_2$	$-L_1$	0	0	0	L_1	L_2
L_3	$-L_1$	0	0	0	L_1	L_2	L_3

Fig. 1 shows the example of two trapping sets that are known to be dominant in the error floor region for regular $d_v = 3$ LDPC codes. The $(5, 3)$ TS is a subgraph consisting of 5 variable nodes and 3 odd-degree check nodes while the $(6, 4)$ TS is a subgraph consisting of 6 variable nodes and 4 odd-degree check nodes. If such structures are contained in the Tanner graph of the code, they cause iterative decoders (BP-based as well as Min-Sum-based) to fail when errors are located in the variable nodes of the trapping sets. The presence of these structures is the source of the error floors for LDPC decoders.

Fig. 1. Examples of trapping sets for regular $d_v = 3$ LDPC codes.

The selection method for FAIDs begins by identifying such harmful trapping sets from the trapping set ontology which is a database of trapping sets with hierarchy established by their topological relations [4], and then analyzing the message passing of a given FAID on each of these isolated structures in order to examine its error correction capability. The influence of an arbitrary neighborhood of the harmful structure during message-passing is partially captured by considering different possible messages that enter the trapping set. Based on the analysis on each trapping set, the FAID with the best error correction capability is chosen. For the design of the FAID defined by Table I, the analysis of the message passing was done on the $(5, 3)$ TS as well as $(6, 4)$. More details on the strategies and algorithms used to design good FAIDs can be found in [12].

Using the above approach, it is possible to design 7-level FAIDs that are capable of outperforming the floating-point BP as well as the quantized Min-Sum-based decoders in the error floor on a given code. Moreover, a single particularly

good 7-level FAID identified from the selection methodology is capable of surpassing BP on several codes.

D. Simulation Results

In order to illustrate the efficiency of FAIDs, Fig. 2 shows the frame error rate (FER) performance comparisons between the BP, the Min-Sum-based decoders, and the 7-level FAID as a function of the cross-over probability α over the BSC on a $(7807, 7177)$ QC-LDPC code with $L = 211$, $r = d_c = 37$, and $t = d_v = 3$. Table I was used as the VNU function Φ_v for the 7-level FAID.

The parameters for the 3-bit offset Min-Sum decoder and the 6-bit offset Min-Sum decoders are $(C = 2, \gamma = 1)$ and $(C = 10, \gamma = 3)$ respectively. Note that the offset correction values and the channel value of the Min-Sum decoders have been adapted to our particular simulation settings, and optimized through a density-evolution analysis by maximizing their decoding threshold for regular $d_v = 3$ codes on the BSC channel, combined with the selection procedure that was proposed for FAIDs based on trapping sets in order to obtain a better performance in the error floor region. To the best of our knowledge, we are not aware of any other decoder designs in the literature that specifically optimize the performance on the BSC.

From the figure, it is evident that the 7-level FAID clearly outperforms the 5-bit Min-Sum (which exhibits very poor performance), the 3-bit offset Min-Sum, and the floating-point BP decoder in the error-floor region. Notice how the 6-bit offset Min-Sum decoder approaches the performance of BP in the waterfall while achieving significantly improved performance in the error floor. For this code, the 7-level FAID with only 3 bits of precision is able to perform close to the 6-bit offset Min-Sum decoder in the error floor region.

Fig. 2. Performance comparisons between various LDPC decoders for a $(7807, 7177)$ QC-LDPC code

Additional simulation results are provided in Fig. 3 to show that the good 7-level FAID depicted in Table I is capable of

surpassing BP on other codes. These results were obtained on a (2388, 1793) QC-LDPC code that was designed for the best BP performance (for that code rate and length) by avoiding certain harmful trapping sets during the code construction [14]. Even for such a code, the 7-level FAID provides superior FER performance in the error floor region compared to the floating-point BP. Note that for this code, the 7-level FAID surpasses the 6-bit offset Min-Sum decoder in the error-floor regions. Hence 7-level FAIDs, which are 3-bit decoders, are capable of outperforming not only the floating-point BP decoders but also 6-bit Min-Sum-based decoders. The number of maximum decoding iterations used for all the decoders is set to 100 in our simulations.

Fig. 3. Performance comparisons between various LDPC decoders for a (2388, 1793) QC-LDPC code

E. Lowering the Error Floor with FAID Diversity

Since a particular FAID rule is designed to avoid the attraction of some specific types of error patterns located on trapping sets, one can think of using different decoding rules that are designed for different error patterns, and combine them in the decoder. This approach has been proposed in [13] under the name of *FAID diversity*. As a consequence, the use of the FAID diversity framework allows to collectively correct a very large number of dominant error patterns, and possibly reach a guaranteed error correction result, when a specific LDPC code is targeted. Let us briefly describe the optimization procedure and the FAID diversity decoder, and for more information, one can refer to [13].

The message passing decoder using a multiplicity of different FAIDs follows a sequential procedure: let us assume that the decoder uses $N_D \geq 2$ FAIDs. For each and every channel noise realization, the FAIDs are run iteratively for a maximum of N_I . When a given FAID $\mathbf{D}_{d=1 \dots N-D}$ fails to converge after N_I iterations, then the channel values are used to re-initialize the next decoder $\mathbf{D}_{d+1}$, until a codeword is found or the maximum number of considered FAIDs has been

reached ($d = N_D$). Note that in the case $\mathbf{D}_d$ converges to a wrong codeword, we declare it as an undetected error, and do not switch to the next decoder. The fact that the FAIDs are used sequentially does not decrease the average throughput compared to the use of a single FAID. Indeed, since in the error floor region the first FAID has already a $\text{FER} < 10^{-6}$, this means that the FAID diversity feature is triggered only for an average of one frame over 10^6 decoded frames. A more precise discussion on the impact of FAID diversity on the hardware realization of the decoder is presented in section V.

In this paper, we took the (7807, 7177) QC-LDPC code as example and our goal has been, not only to improve the error correction performance in the error floor region, but also to target a guaranteed error correction feature of the FAIDs using the concept of diversity. In order to do so, the additional FAIDs rules (apart from the first one described in Table I) will be *code dependent* as designed to specifically correct the dominant low-weight error patterns of the (7807, 7177) QC-LDPC code. Since the $(N = 7807, K = 7177)$ QC-LDPC code we use in this paper has minimum distance $D_{\min} = 10$, we have targeted a guaranteed error correction of $k = 4$ errors. The problematic error patterns of Hamming weight $k = 4$ have been detected using the knowledge of the small Trapping sets in the LDPC code [15], [16] and a set of $N_D = 7$ FAIDs has been selected to correct all the identified error patterns using the selection procedure described in [13]. The list of the $N_D = 7$ FAIDs is given in appendix A (the first decoder is the one of Table I). Although we do not have a proof that the set of $N_D = 7$ FAIDs can guarantee the correction of all 4-error patterns, we rely on the conjecture presented in [13] to have a strong confidence that it is the case. As an additional result, when one uses the $N_D = 7$ FAIDs in a serial manner in the decoding algorithm, the performance in the error floor is greatly increased as we gain a decade in the error floor region compared to the use of a single FAID decoder (see Fig. 4). In this figure, all decoders have been run for a maximum of 100 decoding iterations, and stopped when a codeword was detected. To conclude, the performance of 7-level FAIDs with diversity surpass all other decoders by one to several order of magnitude in the error floor, including the offset-corrected Min-Sum with 6 bits of precision.

In the next section, we will discuss the hardware implementation of FAIDs and provide synthesis results comparing FAIDs with Min-Sum-based decoders. We will show that with the proposed VLSI architectures, FAIDs can also provide substantial gains in chip area.

IV. VLSI ARCHITECTURES FOR FAIDS

In this section, an efficient FAID implementation architecture is developed for QC-LDPC codes. Since the FAID require very short word length, adopting bit-serial CNU and processing all the v-to-c messages connected to a check node simultaneously would lead to higher efficiency. This section proposes an innovative bit-serial CNU architecture based on the design in [10]. The CNU in [10] only computes $\text{Min}1$, and $\text{Min}1+1$ is used as $\text{Min}2$. However, such an approximation could lead to performance loss and raise the error floor in

Fig. 4. Performance of FAID diversity for the (7807, 7177) QC-LDPC code

the FAID. In our proposed CNU, both Min1 and Min2 are computed at the expense of small area overhead. Moreover, the VNU can be efficiently implemented due to the symmetry in the Boolean map table. An optimized interleaved data scheduling scheme is also developed in this section to maximize the hardware utilization efficiency of the FAID.

A. CNU Architecture

Fig. 5. Bit-serial CNU architecture for the FAID

Fig. 5 shows the proposed bit-serial CNU architecture. The d_c v-to-c messages are in sign-magnitude form. They are input simultaneously starting from the most significant bits (MSBs) of the magnitudes, and the sign bits are loaded last as the least significant bits (LSBs). Each input has a flag f_i ($1 \leq i \leq d_c$). $f_i = '0'$ means that the corresponding input is a candidate of the minimum magnitude. At the beginning, the flags are all '0'. After a flag is flipped to '1', it will stay at '1' and mask the corresponding input through the OR gate on the left column of Fig. 5. The outputs of the OR gates are ANDed together. If any unmasked input bit is '0', the output of the d_c -input AND gate in the middle is '0'. As a result, the unmasked inputs whose bits are '1' will have the corresponding flags flipped to '1'. If none of the unmasked inputs is '0', then the flags remain unchanged. Apparently, the output of the d_c -input AND gate is the bit of the minimum magnitude in each clock

cycle. Assume that w is the word length, Min1 is available at the registers after $w - 1$ clock cycles. Also, the index I can be derived from the flags at this time. In the w th clock cycle, the signs of the input messages are XORed to compute S .

In our design, Min2 is computed after Min1 is derived. There are two cases to consider.

Case 1: There is only one unique Min1, and accordingly only one flag is '0' at the end of the $w - 1$ th clock cycle. In this case, Min2 should be the minimum magnitude among the rest $d_c - 1$ inputs. Hence, it can be computed by using the architecture in Fig. 5 for a second round and initializing the flag corresponding to the input that equals Min1 as '1'. The d_c flags for Min1 should be loaded into another set of registers to derive the index I before the second round starts. Also the second round only needs $w - 1$ clock cycles since S does not need to be computed again.

Case 2: There are multiple equal minimum magnitudes in the input messages, and accordingly multiple zero flags at the end of the computation for Min1. In this case, Min2 is equal to Min1, and does not need further computation.

To differentiate these two cases, a unique zero detector (UZD) is included in the CNU. Its output signal, 'd', is asserted when there is only one zero flag. It can be generated as

$$d = \sum_{i=1}^{d_c} f'_i \prod_{j \neq i} f_j,$$

where the add and multiply denote logic OR and AND, respectively. Through substructure sharing, the area requirement of the UZD can be reduced.

B. VNU Architecture

The Boolean map for the VNU in the FAID can be efficiently implemented in a bit-parallel way using logic gates. d_v c-to-v messages and the channel information are available to the VNU at the same time. After one clock cycle, d_v v-to-c messages are derived simultaneously. To facilitate the computations in the CNUs, the alphabet levels are encoded into binary representations according to sign-magnitude format. Since the Boolean map is symmetric, the logic expression of each output bit can be effectively simplified through Karnaugh-map reduction. For the purpose of comparison, Fig. 6 shows the architectures of the VNUs for the Min-Sum decoder and the FAID. Besides computing the v-to-c messages, the VNUs also add up all the c-to-v messages and the channel information. The sign of the sum is the hard-decision. In the VNU of the Min-Sum decoder with $d_v = 3$, seven adders are required to take care of the corresponding computations in one clock cycles. Moreover, the Min-Sum decoder needs saturation modules to bound the sum of the messages within w bits. One more clock cycle is required to implement the saturation. These modules are not necessary in the VNU of the FAID since the output messages are decided by the Boolean map and have the same word length as the input messages. The CNUs need the messages in sign-magnitude format. In the FAID, the inputs and outputs of the boolean map are both in sign-magnitude format. The messages only need to be converted to two's complement format to compute

Fig. 6. VNU architectures: (a) VNU of the Min-Sum decoder; (b) VNU of the FAID

the hard-decisions. On the contrary, the VNUs in the Min-Sum decoder add up the c-to-v messages to derive the c-to-v messages as shown in (3). Therefore, conversion units are needed both at the inputs and outputs.

C. Top Level Decoder Architecture

Fig. 7. Top level architecture of the FAID

The top level architecture of our proposed FAID architecture is shown in Fig. 7. Assume that the H matrix of the QC-LDPC code has $r \times t$ nonzero sub-matrixes of dimension $L \times L$, our proposed decoder employs rL CNUs to process all rows of H simultaneously in order to reach high throughput for optical communication and data storage systems. Accordingly, the check node processing for a decoding iteration can be finished in $2w$ clock cycles: w clock cycles to find $\{Min1, I, S\}$, one clock cycle for unique zero detection, and $w - 1$ clock cycles to compute $Min2$ if necessary. In each decoding iteration, variable node processing follows check node processing. The VNUs and CNUs are connected through permutation networks according to the locations of the nonzero entries in the H matrix. Since the computations of VNUs and CNUs are carried out in bit-parallel and bit-serial manner, respectively, parallel-to-serial (serial-to-parallel) convertors are required before the messages are passed from VNUs to CNUs (CNUs to VNUs). The information $\{Min1, Min2, I, S\}$ computed from the CNUs are stored in registers inside the CNUs. The total size

of the registers is $(2w + \lceil \log 2d_c \rceil + 1) \times r \times L$. Moreover, the hard-decision bits and the channel information are stored in memory blocks associated with the VNUs.

To increase the hardware utilization efficiency, two data blocks can be interleaved as proposed in [10], so that the variable node processing for one data block overlaps the check node processing of the other block. However, in the design of [10], tL VNUs are adopted, and each bit-parallel VNU generates d_c v-to-c messages for the same column of H in one clock cycle. Hence, the variable node processing only takes one clock cycle, which is much shorter than the latency of the check node processing. This causes the VNUs to be idle most of the time. To maximize the hardware utilization efficiency, we propose to adopt less copies of the VNUs, and use the VNUs in a time-multiplexed manner to match the speed of the CNUs. As a result, the hardware cost of the overall decoder can be significantly reduced without sacrificing the throughput.

Fig. 8 shows the proposed scheduling of the computations in the decoding process. The darker and clear bars belong to the decoding of two data blocks. Since each column of H does not have a dedicated VNU in our design, the results of the CNUs need to be permuted before they are sent to the VNUs according to the locations of the nonzero entries in H . This permutation can be done by multiplexors in one clock cycle. To match the speed of the CNUs, the columns of H need to be divided into $2w - 1$ groups, and the variable node processing for each group is completed in one clock cycle. In addition, the permutation can be simplified as barrel shifting if the columns of H processed each time are whole block columns of size L . Therefore, $L[t/(2w-1)]$ VNUs can be adopted. The digits on the bars for the VNUs in Fig. 8 are the indices of the groups of columns. Without sacrificing the throughput, our scheme requires a substantially smaller number of VNUs than that in [10] at the expense of a small permutation network. Due to the interleaving, the registers for storing $\{Min1, Min2, I, S\}$ and the v-to-c messages need to be doubled to avoid access conflicts. These registers are also used as serial-to-parallel and parallel-to-serial converters between the VNUs and CNUs.

V. COMPLEXITY ANALYSIS AND COMPARISONS

Taking a (7807, 7177) QC-LDPC code with $L = 211$, $r = d_c = 37$, and $t = d_v = 3$ as an example, the proposed FAID is synthesized in this section and compared with the Min-Sum decoder architecture in [10], which is among the most efficient existing designs. As shown in Fig. 2 and Fig. 3, with 7-level alphabets, which translate to 3-bit word length,

Fig. 8. Scheduling of the computations in the decoding process

TABLE III
SYNTHESIS RESULTS FOR CNUS USING 180nm CMOS TECHNOLOGY

	Min-Sum CNU [10] ($w = 5$)	Min-Sum CNU [10] ($w = 6$)	Proposed CNU ($w = 3$)
Area (μm^2) (normalized)	6306 (1)	6306 (1)	7662 (1.21)
Max. freq. (Mhz)	384	384	384
Latency (# of clks)	6	7	6

TABLE IV
SYNTHESIS RESULTS FOR VNUS USING 180nm CMOS TECHNOLOGY

	Min-Sum VNU [10] ($w = 5$)	Offset Min-Sum VNU [10] ($w = 6$)	Proposed FAID VNU ($w = 3$)	Proposed 7-Table FAID VNU ($w = 3$)
Area (μm^2) (normalized)	5009 (1)	6370(1.27)	4454 (0.89)	8418 (1.68)
Max. freq. (Mhz) (normalized)	555 (1)	500 (0.90)	625 (1.13)	625 (1.13)
Latency (# of clks)	1	2	1	1

the FAID decoder can achieve better performance than the belief propagation and offset Min-Sum decoder with 6-bit word length in the error floor region. It also outperforms the standard Min-Sum decoder and offset Min-Sum decoder with the same word length in the waterfall region. The proposed CNU and VNU for the FAID with $w = 3$ are synthesized using SMIC 180nm CMOS technology and compared with those with $w = 5$ and 6 for the Min-Sum decoder using the architecture from [10] in Table III and IV, respectively. During the synthesis, the clock frequency is set to higher values gradually, and the largest clock frequency that does not lead to sudden increase in the area is listed as the maximum achievable clock frequency in these tables.

Since the CNU for the Min-Sum decoder in [10] is bit-serial, the gate count remains unchanged for different word length w . The offset operation for the offset Min-Sum decoder is achieved more efficiently in bit-parallel manner. Hence, the corresponding complexity is included in the VNU instead. Compared to the CNU in [10], the extra UZD needed in the proposed CNU causes 21% area increase. The Min-Sum check node processing in [10] with word length w can be finished in w clock cycles. However, another clock cycle is spent on computing $\text{Min}2 = \text{Min}1 + 1$. Hence, the latency of the Min-Sum CNU in [10] is $w + 1$ clock cycles.

It can be seen from Table IV that the proposed FAID VNU has smaller area. The major reason is that multiple 5 or 6-bit adders are required in the VNUs of the Min-Sum decoder as shown in Fig. 6(a), while the 3-bit symmetric Boolean map in the FAID can be implemented with simpler logic. In addition, the VNUs in the Min-Sum decoder need saturation modules and more units to convert messages from two's complement to sign-magnitude format. The 6-bit VNU for the offset Min-Sum decoder is larger than the 5-bit VNU for the standard Min-Sum decoder because of the longer word length and the extra computations needed for the offset. Also the offset calculations

lead to another stage of pipelining latency

The synthesis results of the overall (7807, 7177) QC-LDPC decoders are listed in Table V. Compared to the Min-Sum decoders using the architecture in [10], the number of VNUs needed in our FAID design is reduced to less than 1/4, and each VNU is smaller. Moreover, the registers for storing the v-to-c messages, Min1, and Min2 in the FAID is less since the word length is shorter. Although the FAID requires extra permutation networks and its CNUs have additional UZDs and copies of registers for storing the flags resulted from the Min1 computation, the VNUs dominate the overall decoder area due to the high code rate. As a result, the proposed FAID needs 52% or less of the area of the Min-Sum decoders using the architecture in [10]. After pipelining the VNUs in the offset Min-Sum decoder, the critical paths of all three decoders lie in the CNUs, and they are the same. Although the FAID and standard Min-Sum decoders require the same number of clock cycles in each decoding iteration, the offset Min-Sum decoder needs more clock cycles due to the extra pipelining stage in the VNU. Assuming 15 decoding iterations are needed and two data blocks are interleaved, the proposed FAID decoder can achieve $(2 \times 7807) \times 384 / (15 \times 2 \times 6) = 33.3\text{Gbps}$ throughput. If 100 decoding iterations are required, the throughput is reduced to $(2 \times 7807) \times 384 / (100 \times 2 \times 6) = 5\text{Gbps}$.

The synthesis results are also compared with the designs in [17], which is the one of the most efficient design using Min-sum algorithm. The area of the design in [17] uses 65nm technology and the corresponding result has been scaled to 180nm technology for fair comparison. Moreover, the code length in [17] is slightly longer than the code in this paper. As can be seen, after scaling the process, the proposed design requires 57% area of the proposed design. However, the [17] requires seven times longer decoding latency since it adopts a partial parallel scheduling and leads to much less throughput. Considering this, the proposed decoder achieves

TABLE V
SYNTHESIS RESULTS QC-LDPC DECODERS USING 180nm CMOS TECHNOLOGY

	Min-Sum [10] ($w = 5$)	Offset Min-Sum [10] ($w = 6$)	Min-sum [17] ($w = 4$)	Proposed FAID ($w = 3$)	Proposed 7-Table FAID ($w = 3$)
Code length	7897	7807	9216	7807	7807
Area (mm^2) (normalized)	69.3 (1)	79.8(1.15)	20.8 (0.3)	36.1 (0.52)	44.4 (0.64)
Max. freq. (Mhz)	384	384	300	384	384
Latency (# of clks)	180	210	1262	180	180
Throughput (Gbps) (15 iter.)	33.3	28.5	2.1	33.3	33.3

Fig. 9. Different entries between Table I and those in the appendix

higher efficiency compared to [17].

As discussed in Section III.E, using different decoders one after another improves the performance. The decoders are only different in the tables used for the variable node processing. Hence, only the VNUs need to be modified in order to accommodate multiple decoders. Most of the entries in Table I and the six tables listed in Table VI in the appendix are the same. As a result, many logic gates can be shared to implement the seven tables. Fig. 9 shows the different entries of the tables. This figure has 7×7 blocks denoting the 7×7 entries in a table. If a table has an entry different from that in Table I, a dot is put into the corresponding block. Dots of different colors correspond to different tables. The dots in this figure only indicate the locations of the differences. However, they facilitate the discovery of the groups of tables among D2 to D7 that share the same entries. For example, the entries for $l_{1,7}$, $l_{3,7}$, $l_{7,1}$ and $l_{7,3}$ in Fig. 9 have the same four colored dots and tell that D3, D4, D5, and D7 are different from Table I in these entries. From Table VI, it can be also observed that the these four entries in the four tables are actually the same, and hence the corresponding logic can be shared in the implementations of D3, D4, D5, and D7. Similarly, by observing the other blocks in Fig. 9 that have the same colored dots and then referring to Table VI, the other terms sharable in the implementation of different tables are located. The synthesis results of the VNU implementing 7 tables are listed in the last column of Table IV. The VNU implementing 7 tables is set with the same timing constraint as the one-table counterpart. As can be seen in Table IV, only 89% extra area is required when 7 tables are all involved. The major reason is that by taking advantage of the characteristics of the 7 tables, many logic gates can be shared and thus many cases can be

grouped together for the 7-table VNU instead of implementing 7 individual tables separately. The corresponding result for overall decoder implementing 7 tables is also included in Table V. Compared to the one-table counterpart, only 23% extra area is needed.

VI. CONCLUSION

In this paper, a low-complexity implementation architecture was developed for the recently introduced FAIDs. A novel bit-serial CNU was designed for the FAID and the implementation of the Boolean map for VNU is simplified using its symmetric property. In addition, the computation scheduling was optimized to fully utilize the hardware units. Compared to most efficient existing Min-Sum decoders, the proposed FAID decoder requires substantially smaller area to achieve the same throughput. Our future work will be devoted to two aspects. First, we will continue to reduce the complexity of multiple FAIDs with different Boolean maps, which can be adopted to further lower the error floor. Another interesting direction of research includes the generalization of the FAID framework for larger alphabets, other LDPC codes distributions, or more complex channels. Neither of those generalizations are trivial because of the following reason: our approach of decoder design using FAIDs relies on the exact knowledge of the weaknesses of a particular LDPC code, decoded with FAID. Those weaknesses for regular $d_v = 3$ codes and the BSC channel are known as they are precisely defined as the trapping sets. For a generalization of FAIDs, one then has to characterize the weaknesses of $d_v = 4$ or even irregular LDPC codes on the AWGN channel, but we know that it is not just a simple extension of the trapping-set analysis, as it is related to the more complicated notion of pseudo-codewords.

APPENDIX A

LIST OF 7-LEVEL FAIDS USED IN THE FAID DIVERSITY ALGORITHM

In Table VI, we list the 7-level FAIDs used in this paper for the decoding of the (7807, 7177) QC-LDPC code. We only indicate the entries $l_{i,j}$ of the LUT array (see Table I) that cannot be deduced by symmetry. In this table, $l_{i,j}$ represents the LUT entry in the i -th row and the j -th column. Note that the first decoder D0 in table VI is the one indicated in the LUT of Table I.

REFERENCES

- [1] M. Davey and D. J. MacKay, "Low density parity check codes over GF(q)," *IEEE Commun. Letter*, vol. 2, pp. 165-167, Jun. 1998.

TABLE VI

LIST OF THE 7-LEVEL FAIDS USED IN THIS PAPER. THE SEVEN FAIDS ARE CONJECTURED TO GUARANTEE AN ERROR CORRECTION OF $t = 4$ ON THE (7807, 7177) QC-LDPC CODE.

FAID	$l_{1,1}$	$l_{1,2}$	$l_{1,3}$	$l_{1,4}$	$l_{1,5}$	$l_{1,6}$	$l_{1,7}$	$l_{2,2}$	$l_{2,3}$	$l_{2,4}$	$l_{2,5}$	$l_{2,6}$	$l_{2,7}$	$l_{3,3}$	$l_{3,4}$	$l_{3,5}$	$l_{3,6}$	$l_{3,7}$	$l_{4,4}$	$l_{4,5}$	$l_{4,6}$	$l_{4,7}$	$l_{5,5}$	$l_{5,6}$	$l_{5,7}$	$l_{6,6}$	$l_{6,7}$	$l_{7,7}$
D_0	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	0	L_2	$-L_1$	0	L_1	L_2	0	L_1	L_2	L_2	L_3	L_3
D_1	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_1$	L_1	$-L_2$	$-L_1$	$-L_1$	0	L_2	$-L_1$	0	L_1	L_2	L_1	L_1	L_2	L_2	L_3	L_3
D_2	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	L_1	$-L_2$	$-L_1$	$-L_1$	$-L_1$	L_1	$-L_1$	$-L_1$	L_0	L_2	0	L_1	L_2	L_1	L_2	L_3
D_3	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_2$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_2$	L_1	$-L_3$	$-L_2$	$-L_1$	0	L_1	$-L_1$	0	0	L_3	0	L_1	L_3	L_1	L_3	L_3
D_4	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_1$	L_1	$-L_2$	$-L_2$	$-L_1$	0	L_1	$-L_1$	0	L_1	L_1	L_1	L_1	L_1	L_1	L_3	L_3
D_5	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	0	$-L_3$	$-L_3$	$-L_2$	$-L_2$	$-L_2$	L_1	$-L_2$	$-L_2$	$-L_1$	0	L_2	$-L_1$	0	0	L_2	L_1	L_1	L_2	L_1	L_2	L_2
D_6	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_3$	$-L_1$	$-L_3$	$-L_3$	$-L_3$	$-L_2$	$-L_2$	L_1	$-L_2$	$-L_1$	$-L_1$	0	L_1	$-L_1$	0	0	L_2	0	L_1	L_2	L_1	L_2	L_3

- [2] J. Chen, *et. al.*, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Commun.*, vol. 53, pp. 1288-1299, Aug. 2005.
- [3] T. Richardson, "Error floors of LDPC codes," *Proc. 41st Annual Allerton Conf on Communications Control and Computing*, 2003.
- [4] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," *Proc. 47th Annual Allerton Conf. on Commun., Control, and Computing*, Sep. 2009.
- [5] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. on Commun.*, vol. 57, no. 6, pp.1663-1673, Jun. 2009.
- [6] S. K. Planjery *et. al.*, "Iterative decoding beyond belief propagation," *Proc. Info. Theory and Appl. Workshop*, Feb. 2010.
- [7] J. Kang, Q. Huang, S. Lin, and K. Abdel-Ghaffar, "An iterative decoding algorithm with backtracking to lower the error-floors of LDPC codes," *IEEE Trans. on Commun.* vol. 59, no. 1, pp. 64-73, Jan. 2011.
- [8] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasic, "Finite alphabet iterative decoders for LDPC codes surpassing floating-point iterative decoders," *Electronics Letters*, vol. 47, no. 16, pp. 919-921, Aug. 2011.
- [9] S. K. Planjery, D. Declercq, S. Chilappagari, and B. Vasic, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," *Proc. IEEE Intl. Symp. Info. Theory*, pp. 769-773, Austin, TX, Jul. 2010.
- [10] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A bit-serial approximate Min-Sum LDPC decoder and FPGA implementation," *Proc. IEEE Intl. Symp. Circuits and Syst.*, May 2006.
- [11] F. Cai, X. Zhang, D. Declercq, B. Vasic, D. V. Nguyen and S. K. Planjery, "Low-complexity finite alphabet iterative decoders for LDPC codes," *Proc. IEEE Intl Symp. Circuits and Syst.*, May 2013.
- [12] S.K. Planjery, D. Declercq, L. Danjean and B. Vasic, "Finite Alphabet Iterative Decoders, Part I: Decoding Beyond Belief Propagation on the BSC", *IEEE Trans. on Commun.*, vol. 61, no. 10, pp. 4033-4045, Oct. 2013.
- [13] D. Declercq, B. Vasic, S.K. Planjery and E. Li, "Finite Alphabet Iterative Decoders, Part II: Towards Guaranteed Error Correction of LDPC Codes via Iterative Decoder Diversity", *IEEE Trans. on Commun.*, vol. 61, no. 10, pp. 4046-4057, Oct. 2013.
- [14] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280-2302, Apr. 2012.
- [15] D. Declercq and M. Fossorier, "Improved impulse method to evaluate the low-weight profile of sparse binary linear codes," *Proc. IEEE Int. Symp. Inform. Theory*, Toronto, Canada, pp. 1963-1967, Jul. 2008.
- [16] S. Abu-Surra, D. Declercq, D. Divsalar, and W. Ryan, "Trapping set enumerators for specific LDPC codes," in *Proc. Inform. Theory App. Workshop*, San Diego, CA, pp.1-5, Feb. 2010.
- [17] H. Zhong *et al.* "Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording," *IEEE Transactions on Magnetics*, vol. 43, no. 12, pp. 4117-4122, Dec. 2007.