Low-complexity Finite Alphabet Iterative Decoders for LDPC Codes

Fang Cai, Xinmiao Zhang
Case Western Reserve University
{fang.cai, xinmiao.zhang}@case.edu

David Declercq
ETIS Laboratory
declercq@ensea.fr

Bane Vasic, Dung Viet Nguyen, and Shiva Planjery
University of Arizona
{vasic, nguyendv,shivap}@ece.arizona.edu

***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. Recently, a new type of decoders termed finite alphabet iterative decoders (FAIDs) were introduced. The FAIDs use simple Boolean maps for variable node processing. With very short word length, they can surpass the BP-based decoders in the error floor region. This paper develops a low-complexity implementation architecture for FAIDs by making use of their properties. Particularly, an innovative bit-serial check node unit is designed for FAIDs, and the symmetric Boolean maps for variable node processing lead to small silicon area. An optimized data scheduling scheme is also 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 Min-sum decoders for an example (7807, 7177) LDPC code, while achieving better error-correcting performance 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. The BP-based decoding can approach the Shannon limit. Nevertheless, due to the presence of cycles in the corresponding Tanner graph, the error-correcting performance curve in high signal-to-ratio (SNR) region can flatten out. This is called the error floor. It happens because the decoding can get 'stuck' in trapping sets [3], [4] and fail to converge 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 rate, 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, finite alphabet iterative decoders (FAIDs), were introduced for LDPC codes [8]. In these decoders, the messages are represented by alphabets with a very limited 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. It has been shown that, with only seven levels in the alphabets, which translate to 3-bit word length, FAIDs can outperform floating-point BP decoders in the error-floor region over binary symmetric channel. In addition, multiple FAIDs with different map functions can be adopted to further improve the performance at the cost of higher complexity [9].

This paper proposes a low-complexity implementation architecture for FAIDs by making use of their properties. Since the required word length is very short, adopting bit-serial check node units (CNUs) and processing all the v-to-c messages connected to a check node simultaneously would lead 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 as the Min-sum decoder in [10], while achieving better performance at the error-floor region.

The structure of this paper is as follows. Section II introduces FAIDs. The proposed decoder architectures are presented in Section III. Section IV analyzes the hardware complexity and provides comparisons with Min-sum decoders, and conclusions are drawn in Section V.

II. FAIDs and Min-sum Decoders

An LDPC code is a linear block code that can be defined by the corresponding parity check matrix H or the associated Tanner graph. In the Tanner graph, 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 hardware implementations.

This material is based upon work supported by the National Science Foundation under Grants No. 0846331, 0802159, and CCF-0963726

TABLE I
BOOLEAN MAP FOR FAID VARIABLE NODE PROCESSING WITH 7-LEVEL ALPHABETS WHEN $y = -C$

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

In FAIDs and BP-based decoders, messages are passed iteratively between check and variable nodes until a valid codeword is found or the maximum iteration number is reached. Each message in FAIDs is represented by an alphabet that is confined to $2s+1$ levels $\{0, \pm L_i : 1 \leq i \leq s\}$, where $L_i \in \mathbb{R}^+$ and $L_i > L_j$ for any $i > j$. Moreover, the message from the channel is denoted by y, and $y = \pm C$ for binary symmetric channel. Let $m_1, \ldots, m_{l-1}$ represent the incoming messages to a node with degree l. In this paper, we focus on codes with constant variable node degree $d_v = 3$. VNUs compute v-to-c messages. The v-to-c message to a check node is derived based on the channel information and the c-to-v messages from all other connected check nodes computed in the previous decoding iteration. An example of a Boolean map, Φ_v, for computing the v-to-c messages in a FAID with 7-level alphabets is shown in Table I when $y = -C$. The table for $y = C$ is symmetric to Table I. On the other hand, the function Φ'_v used in the VNUs of Min-sum decoders is

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

Moreover, for both decoders, the hard-decision of each bit is updated as the sign of $y + \sum_{j=1}^{d_v} m_j$ in each decoding iteration.

Both FAIDs and Min-sum decoders share the same check node processing, which computes c-to-v messages. The c-to-v message to a variable node is calculated using the v-to-c messages from all other connected variable nodes. Assume the check node degree is d_c, the function at the CNU, Φ_c, is

$$\Phi_c(m_1, \ldots, m_{d_c-1}) = \prod_{j=1}^{d_c-1} sign(m_j) \min_{j \in 1,\ldots,d_c-1} (|m_j|). \quad (2)$$

From (2), the c-to-v messages from a check node can only have two possible magnitudes, the minimum and second minimum ones among the magnitudes of all incoming v-to-c messages. They are denoted by Min1 and Min2, respectively. As a result, only four values need to be recorded for each check node: Min1, Min2, $S = \prod_{j=1}^{d_c} sign(m_j)$, 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 those 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.

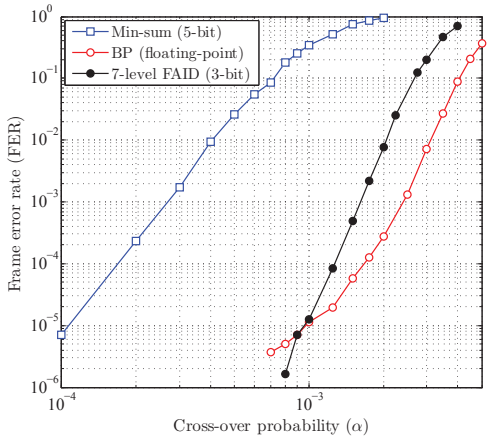

Fig. 1. Performance of LDPC decoders for a (7807, 7177) QC-LDPC code

A complete treatment of the FAID can be found in [8]. The frame error rate (FER) of the FAID for a (7807, 7177) QC-LDPC code with $L = 211$, $r = d_c = 37$, and $t = d_v = 3$ over binary symmetric channel with cross-over probability α and 15 decoding iterations are shown in Fig. 1. As it can be observed, the FAID with 7-level alphabets has better performance than the floating-point BP in low FER region. In addition, it can substantially outperform the Min-sum decoding with 5-bit word length.

III. 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 Min1, and Min1+1 is used as Min2. However, such an approximation could lead to performance loss and raise the error floor in 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.

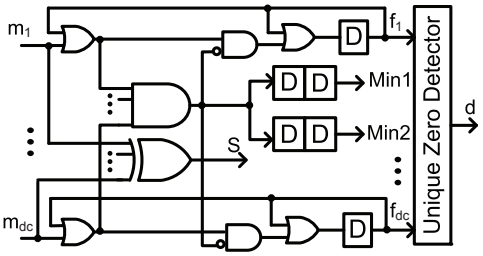

Fig. 2. Bit-serial CNU architecture for the FAID

Fig. 2 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. 2. 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 wth 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. 2 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.

The VNU of 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 according to the Boolean map. 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.

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. 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. 3 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, $L\lceil t/(2w-1) \rceil$ VNUs can be adopted. The digits on the bars for the VNUs in Fig. 3 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.

IV. Complexity Analysis and Comparisons

TABLE II
Synthesis results for CNUs using $180nm$ CMOS Technology

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

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 in [10], which is among the most efficient existing designs. As shown in Fig. 1, with 7-level alphabets,

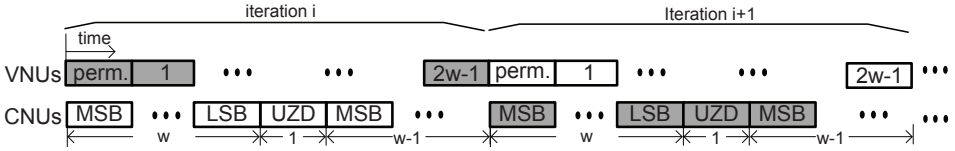

Fig. 3. Scheduling of the computations in the decoding process

TABLE III
SYNTHESIS RESULTS FOR VNUS USING $180nm$ CMOS TECHNOLOGY

	Min-sum VNU [10] ($w = 5$)	Proposed FAID VNU ($w = 3$)
Area (μm^2) (normalized)	5009 (1)	4454 (0.89)
Max. freq. (Mhz) (normalized)	555 (1)	625 (1.13)
Latency (# of clks)	1	1

which translate to 3-bit word length, the FAID decoder can achieve better performance than the Min-sum decoder with 5-bit word length in the error floor region. The proposed CNU and VNU for the FAID with $w = 3$ are synthesized using SMIC $180nm$ CMOS technology and compared with those with $w = 5$ for the Min-sum decoder from [10] in Table II and III, 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.

Compared to the CNU for the Min-sum decoder in [10], the extra UZD needed in the proposed CNU causes 21% area increase. The check node processing for the Min-sum decoder with $w = 5$ can be finished in five clock cycles. However, another clock cycle is spent on computing Min2=Min1+1 in the design of [10]. Hence, the latency of the CNU for the Min-sum decoder with $w = 5$ is also six clock cycles.

It can be seen from Table III that the proposed FAID VNU has smaller area. The major reason is that 5-bit multi-input adders are required in the VNU of the Min-sum decoder according to (1), while the 3-bit symmetric Boolean map in the FAID can be implemented with simpler logic. Moreover, the Min-sum decoder also needs a saturation module to bound the sum of the messages within w bits. This module is not necessary in the FAID VNU since the output messages are decided by the Boolean map.

TABLE IV
SYNTHESIS RESULTS OF (7807, 7177) QC-LDPC DECODERS USING $180nm$ CMOS TECHNOLOGY

	Min-sum [10] ($w = 5$)	Proposed FAID ($w = 3$)
Area (mm^2) (normalized)	69.3 (1)	36.1 (0.52)
Max. freq. (Mhz)	384	384
Latency (# of clks)	180	180
Throughput (Gbps) (15 iter.)	33.3	33.3

The synthesis results of the overall (7807, 7177) QC-LDPC decoders are listed in Table IV. Compared to the Min-sum decoder 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 only needs 52% the area of the Min-sum decoder in [10]. The critical paths of both decoders lie in the CNUs, and they are the same. Also both decoders require the same number of clock cycles in each decoding iteration. Hence, their achievable throughputs are the same. Assuming that 15 decoding iterations are carried out, the latency of both decoders is $15 \times (2 \times 6) = 180$ clock cycles. Considering that two data blocks are interleaved, the achievable throughput is $(2 \times 7807) \times 384/180 = 33.3$Gbps.

V. CONCLUSION

In this paper, a low-complexity implementation architecture was developed for the FAID. A novel bit-serial CNU was designed for the FAID and the implementation of the Boolean map for VNU is simplified using it 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 reducing the complexity of multiple FAIDs with different Boolean maps, which can be adopted to further lower the error floor.

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.
[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 Comuting*, 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. 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. 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, July 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.