An Area-Efficient Relaxed Half-Stochastic Decoding Architecture for Nonbinary LDPC Codes

Xin-Ru Lee, Chih-Wen Yang, Chih-Lung Chen, Hsie-Chia Chang, and Chen-Yi Lee, *Member, IEEE*

***Abstract*—This brief presents an area-efficient relaxed half-stochastic nonbinary low-density parity-check (NB-LDPC) decoder. A novel decoding algorithm, namely, cumulative tracking forecast memory with concealing channel values (CTFM-CC) is proposed to reduce algorithm complexity and maintain bit-error-rate performance as well. Furthermore, the hardware complexity of variable node units (VNUs) is reduced through a truncated architecture, which only keeps the most reliable n probability density functions. To deal with the sum-product-algorithm-to-stochastic conversion of VNU, a dynamic random number generation method, which is used for sampling a stochastic symbol, is also proposed. With these features, a (168, 84) regular-(2,4) NB-LDPC code over GF(16) decoder is implemented in a 90-nm process. According to the results of postlayout simulation, this decoder can deliver a throughput of 1.13 Gb/s with a hardware efficiency of 0.90 Mb/s/K-gate at 286 MHz. Compared to related rate-1/2 NB-LDPC decoders, the proposed decoder achieves the highest hardware efficiency with similar error-correcting capability.**

***Index Terms*—Nonbinary low-density parity-check (LDPC) codes, relaxed half-stochastic (RHS) algorithm, stochastic decoding.**

I. Introduction

NONBINARY low-density parity-check (NB-LDPC) codes, investigated by Davey and Mackay [1], are defined on the null space of a parity-check matrix $\mathbf{H}$, in which the nonzero entries are the elements of a Galois field GF$(q = 2^p)$. Compared to binary LDPC codes, it has been shown that NB-LDPC codes can achieve better bit-error-rate (BER) performance for higher order modulation schemes or multi-input–multi-output communication systems [2]. Despite its remarkable decoding capability, the complicated computational units and huge memory usage are main challenges to implement an NB-LDPC decoder. In this respect, a large amount of literatures with hardware-oriented algorithms have been reported. Declercq and Fossorier [3] proposed the fast Fourier transform (FFT) algorithm to transfer the complicated convolution operations into simpler multiplications. The extended min-sum (EMS) algorithm [4] was proposed to simplify check node unit (CNU) and truncate message vectors from field size q to a limited number n_m. In [5]–[8], the min-max decoding algorithm was introduced to further reduce the complexity of CNU with acceptable performance degradation. On the other hand, the authors in [9] and [10] showed a simplified serial algorithm for a generalized bit-flipping algorithm and a message compression algorithm to reduce the complexity and storage resources required by the variable node unit (VNU).

Stochastic computation is a promising decoding method for error control codes. It is a bit-serial representation of probability and has a great potential to reduce complicated computation. Due to its sequential property, a single wire and simple logic gates can be used to manipulate multiple bits arithmetic. The state-of-the-art works provide high-throughput and area-efficient binary stochastic LDPC decoders for IEEE 802.3an applications [11], [12]. In recent years, stochastic decoding provides alternative methods for NB-LDPC codes [13], [14]. However, more than thousands of decoding cycles in the shift-register edge memory (SR-EM) and tracking forecast memory (TFM) algorithms remain a bottleneck for high-throughput decoders. The relaxed half-stochastic (RHS) algorithm provides higher convergence speed, but it is more complicated than the SR-EM or TFM algorithm due to the conversion of two domains. A field-programmable gate array (FPGA) implementation based on the adaptive multiset stochastic algorithm (AMSA) [14] is introduced to reduce runtime and area cost, but the hardware complexity is still high due to a large amount of memory. Nowadays, pursuing a high-throughput and low-cost decoder is still a design challenge of stochastic NB-LDPC decoding.

In this brief, we propose an area-efficient RHS decoding architecture for NB-LDPC codes. A cumulative TFM with concealing channel values (CTFM-CC) algorithm is proposed to reduce the operations and maintain BER performance. A truncated TFM architecture, as well as its updating criterion, is designed to achieve lower complexity of VNUs. A dynamic random number generation method is provided to generate an output symbol of VNU. The organization of this brief is as follows: In Section II, the background of stochastic decoding for NB-LDPC codes is briefly introduced. Section III presents the proposed stochastic NB-LDPC decoder. Implementation results and comparison are discussed in Section IV. Finally, the conclusion is given in Section V.

II. RHS Decoding of Nonbinary LDPC Codes

A stochastic NB-LDPC decoder consists of three types of processing units, including VNUs, CNUs, and permutation node units (PNUs), as shown in Fig. 1. It is noted that the

Manuscript received August 27, 2014; revised October 10, 2014; accepted November 1, 2014. Date of publication November 7, 2014; date of current version March 1, 2015. This work was supported in part by the Ministry of Science and Technology under Grants MOST 103-2221-E-009-198-MY3 and in part by the Ministry of Economic Affairs under MOEA 101-EC-17-A-01-S1-180 and in part by the MediaTek-NCTU Research Center, National Chiao Tung University. This brief was recommended by Associate Editor S. Palermo.

The authors are with the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: mark@si2lab.org; hcchang@mail.nctu.edu.tw; cylee@si2lab.org).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2014.2368616

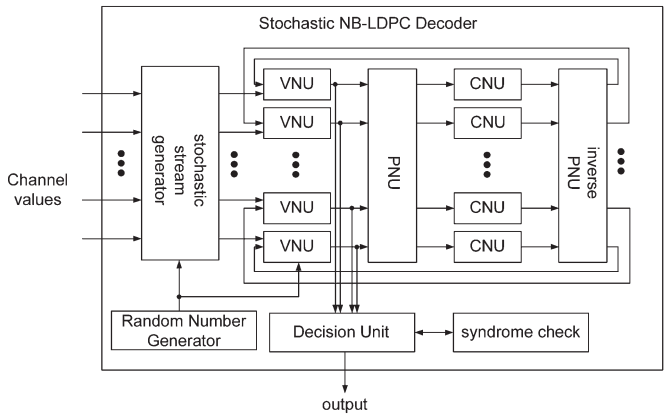

Fig. 1. Architecture of stochastic NB-LDPC decoder.

message representation in the stochastic NB-LDPC decoder is a symbol over GF(q) rather than a probability vector of length q. Therefore, received channel values are converted to stochastic symbol streams over GF(q). The occurrence of a specific element in the stochastic symbol stream is equal to the probability of symbol to be converted. The location of elements in the stream is not important; in other words, the representation of the stochastic symbol stream is not unique. For example, the streams $\{0, 1, 1, \alpha, 0, \alpha^2, \alpha, \alpha^2\}$ and $\{1, \alpha, 0, 1, 0, \alpha^2, \alpha, \alpha^2\}$ are the same for $L_v[0] = 0.25$, $L_v[1] = 0.25$, $L_v[\alpha] = 0.25$, $L_v[\alpha^2] = 0.25$ over GF(4), where $L_v[\gamma]$ is the probability of symbol and $\gamma \in$ GF(4).

In the early research of stochastic LDPC decoding, the main challenge is performance degradation. The reason is the hold state problem, which refers to a case where a set of variable nodes are stuck to a fixed state [11]. Therefore, variable nodes would employ memory devices to track previous symbols and create the random symbol z. The SR-EM and TFM algorithms were proposed to accurately generate a symbol [13]. However, it is hard to implement these algorithms for practical applications due to long decoding cycles. To enhance the rate of decoding convergence, the RHS algorithm for stochastic decoding of NB-LDPC codes was proposed [13]. It was a hybrid decoding technique such that VNUs are operated in the probability domain, but CNUs are operated in the stochastic domain. In other words, the messages to be computed are probabilities in the VNUs, but they are symbols in the CNUs. Compared to other related stochastic works, the number of iterations in an RHS decoder is significantly decreased. Moreover, the complexity of CNUs in a sum-product algorithm (SPA) decoder can be reduced by the stochastic approach. In the following subsections, the functional blocks and the RHS algorithm are described, where the notations are the same with those in [3] except an overline for a stochastic message.

A. CNU

The complexity of CNUs is the highest in the SPA decoder because of the convolution operations. Since the check equations of the RHS algorithm are operated in the symbol domain, the convolution operators can be simplified to finite field summations. Therefore, a CNU is implemented by EXCLUSIVE-OR gates.

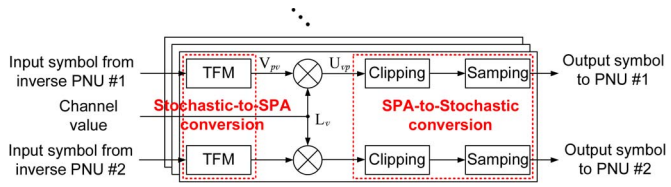

Fig. 2. Architecture of VNU in the RHS algorithm.

B. PNU

In the RHS algorithm, a PNU is fulfilled by a finite field multiplier since the output of PNU equals the product of the input and the corresponding nonzero element in $\mathbf{H}$. In general, a lookup table and a combination of XOR gates is used for the small and high-order field, respectively, since the table size is proportional to the field size.

C. VNU

Fig. 2 shows the architecture of VNU. Since VNUs are operated in the probability domain but inputs/outputs are still symbols, two conversions are required. The first one is a stochastic-to-SPA message conversion. It is performed by a successive-relaxation method [13], in which the probability mass function (PMF) of each symbol γ is stored in TFMs. This updating rule is identical to that in the TFM algorithm. For an incoming message x, the PMFs are updated using the following equation:

$$V_{pv}^{(t)}[\gamma] = \begin{cases} (1-\beta)V_{pv}^{(t-1)}[\gamma] + \beta, & \gamma = x \\ (1-\beta)V_{pv}^{(t-1)}[\gamma], & \text{otherwise} \end{cases} \quad (1)$$

where $V_{pv}^{(t)}[\gamma]$ is the PMF of symbol $\gamma \in$ GF(q), and $\beta < 1$ is a relaxation factor.

After the stochastic-to-SPA message conversion, the VNU v computes a temporal message U_{vp} based on all other PMFs V_{pv} and channel message L_v connected to v as (2), where $\times$ is the term-by-term product of vectors, as follows:

$$U_{vp}^{(t)} = L_v \times \prod_{z=1, z\neq p}^{d_v} V_{zv}^{(t)}. \quad (2)$$

To generate stochastic symbols, the SPA-to-stochastic message conversion will be activated right after (2). Since the final result of cumulation c_q is between 0 and 1, a cumulative density function (CDF) $c_l = \sum_{k=1}^{l} U_{vp}^{(t,k)}, l = 1, 2, \ldots, q$ should be calculated then normalized, such that $c_q = \sum_{k=1}^{q} U_{vp}^{(t,k)} = 1$, where k means the kth symbol of the temporal message U_{vp}. Finally, sampling a stochastic symbol is performed by comparing a uniform random number R to CDF c_l. The output $\overline{U}_{vp}^{(t)}$ is the first symbol $\gamma \in$ GF(q) that its CDF is larger than random number R.

III. Proposed Stochastic NB-LDPC Decoder

Although the RHS algorithm reduces the number of iterations compared to other related stochastic works and alleviates the complexity compared to the SPA decoder, it is necessary to further reduce computations for real applications. For example, VNUs involve term-by-term multiplications to update $U_{vp}^{(t)}$ messages in (2); therefore, area-consuming multipliers should be implemented to finish the operations. Here, we introduce a

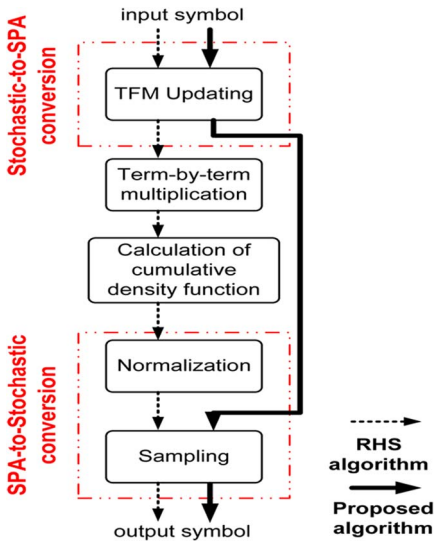

Fig. 3. Decoding process of proposed VNU.

hardware-efficient stochastic NB-LDPC decoder. Fig. 3 summarizes the decoding process of the proposed VNU. The details will be discussed in the following subsections.

A. CTFM-CC

We propose a novel decoding algorithm called CTFM-CC. The feature of CTFM-CC is that the term-by-term multiplications and the computations of CDFs of the RHS algorithm are eliminated, so that it is possible to achieve low complexity of VNUs for NB-LDPC decoders. The CTFM-CC modifies the stochastic-to-SPA message conversion of the RHS algorithm, while retaining the SPA-to-stochastic message conversion as in [13]. The main idea of CTFM-CC comes from the computation of CDFs. Let us observe the following equation, where $i, j \in \mathrm{GF}(q)$ and j is equal to the incoming message x:

$$\begin{aligned} c_l &= \sum_{k=1}^{l} U_{vp}^{(t,k)} = \sum_{k=1}^{l} L_v \times V_{pv}^{(t,k)} \\ &= \ldots + L_v[i] \times V_{pv}^{(t)}[i] + L_v[j] \times V_{pv}^{(t)}[j] \\ &= \ldots + \left(L_v[i] \times (1-\beta) V_{pv}^{(t-1)}[i]\right) \\ &\quad + \left(L_v[j] \times \left((1-\beta) V_{pv}^{(t-1)}[j] + \beta\right)\right) \\ &= \ldots + (1-\beta) \times \underline{\left(L_v[i] \times V_{pv}^{(t-1)}[i] + L_v[j] \times V_{pv}^{(t-1)}[j]\right)} \\ &\quad + \underline{L_v[j] \times \beta}. \end{aligned}$$

The previous equation is similar to the successive-relaxation method, as shown in (1), except the value with an underline. Therefore, in the proposed CTFM-CC, TFMs store and update the results of $L_v[\gamma] \times V_{pv}^{(t-1)}[\gamma]$ instead of $V_{pv}^{(t-1)}[\gamma]$ only. Namely, the effect of channel values are concealed in the TFMs. The computation of CTFM-CC does not require any multiplier since the term-by-term multiplication of (2) disappears. Moreover, to eliminate the calculation of CDF c_i in each decoding cycle, the TFMs of the proposed algorithm store and update cumulative values rather than individual probability of each symbol. As a result, the TFMs are initialized as $\sum_{k=1}^{l} L_v^2[\gamma]$ in place of $L_v[\gamma]$. The updating criterion of the TFMs is the same as (1).

TABLE I
ILLUSTRATION OF THE PROPOSED ALGORITHM

		$V_{pv}^{(t)}[0]$	$V_{pv}^{(t)}[1]$	$V_{pv}^{(t)}[\alpha]$	$V_{pv}^{(t)}[\alpha^2]$
step 1	RHS [13]	0.5	0.3	0.1	0.1
	CTFM-CC	0.25	0.34	0.35	0.36
step 2	RHS [13]	0.4375	0.3875	0.0875	0.0875
	CTFM-CC	0.21875	0.335	0.34375	0.3525
step 3	RHS [13]	0.21875	0.11625	0.00875	0.00875
	CTFM-CC	-	-	-	-
step 4	RHS [13]	0.21875	0.335	0.34375	0.3525
	CTFM-CC	-	-	-	-
step 5	RHS [13]	0.21875	0.335	0.34375	0.3525
	CTFM-CC	0.21875	0.335	0.34375	0.3525

[1] step 1: Initialization, step 2: TFM updating, step 3: Term-by-term multiplication, step 4: Calculation of CDF, step 5: output probability
[2] Assumed $L_v[0] = 0.5$, $L_v[1] = 0.3$, $L_v[\alpha] = 0.1$, $L_v[\alpha^2] = 0.1$ over GF(4), and incoming message x equals to 1, β=0.125

Table I demonstrates an example of how to reduce the operations of VNUs by using the proposed CTFM-CC technique. The initialization of TFMs is accomplished by a multiply-accumulate operation; in other words, the value of each TFM equals to the summation of precedent TFM and square value of channel probability. After that, TFMs are updated based on (1) during decoding cycles. Since the channel values are concealed in TFMs, the multiplication and CDF computation can be eliminated from the operation of VNUs. As shown in step 5 of Table I, the final results of the proposed CTFM-CC and the RHS algorithm are identical.

B. Truncated TFM Architecture

In the conventional RHS decoder, there are q set of registers in each edge of VNUs to keep all the PMFs of q symbols. Namely, the complexity of VNUs is proportional to field size q. To achieve higher area efficiency, the storage requirement of TFMs should be reduced. Based on the experimental results, the symbols can be classified to two categories: only $n (n \ll q)$ symbols with larger PMFs are the candidates of the output symbol, and the other $q - n$ symbols with smaller PMFs can be ignored. For this reason, the truncated TFM with n set of registers to keep the n-largest PMFs is proposed. Therefore, the complexity of VNUs can be reduced by q/n times compared to the conventional RHS decoder.

Different from the TFM algorithm [13], the status of an edge is not considered in this work. Assuming that x is the input symbol and S_{TFM} is a symbol set of a truncated TFM, the truncated TFM is directly updated as follows. First, x is checked whether it is included in the S_{TFM} or not. The truncated TFM is updated based on (3) in case that S_{TFM} contains x, denoted by $x \in S_{\mathrm{TFM}}$. Otherwise, the symbol y with the smallest PMF in the truncated TFM is dropped then x is put into the S_{TFM}. In addition, the PMF of x remains the same with the smallest PMF. As a result, the updating criterion of the truncated TFM is summarized in (3), where $\gamma \in \mathrm{GF}(q)$ and $l = 1, 2, \ldots, n$ is the lth TFM.

The truncated length n is a design parameter, which affects the critical path and area requirement of the decoder. Fig. 4 depicts the BER performance of the proposed decoder under different truncated lengths. It can be observed that using $n = 4$ in the PMFs achieves an acceptable BER performance loss and

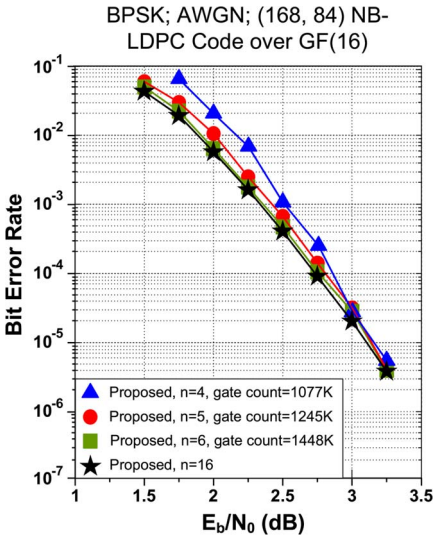

Fig. 4. BER performance and gate count of different truncated lengths.

efficient hardware cost. This truncated length is used in our implementation and comparison. That is

$$\mathrm{TFM}_l^{(t)}[\gamma] = \begin{cases} \mathrm{TFM}_l^{(t-1)}[\gamma], & \text{if } x \notin S_{\mathrm{TFM}} \\ (1-\beta)\mathrm{TFM}_l^{(t-1)}[\gamma], & \text{if } x \in S_{\mathrm{TFM}} \ \& \ \gamma \neq x \\ (1-\beta)\mathrm{TFM}_l^{(t-1)}[\gamma] + \beta * L_v[x], & \text{otherwise} \end{cases}$$

$$S_{\mathrm{TFM}} = \begin{cases} S_{\mathrm{TFM}}, & \text{if } x \in S_{\mathrm{TFM}} \\ \{S_{\mathrm{TFM}} \setminus y\} \cup x, & \text{otherwise.} \end{cases} \quad (3)$$

C. Sampling Unit

During the SPA-to-stochastic conversion of VNU, a stochastic symbol is generated from the truncated TFM according to the PMF values. We use the transformation method [15] as in the RHS algorithm. However, the maximum value of the truncated TFM, which is between 0 and 1, is not a constant in each decoding cycle. The method of the proposed CTFM-CC algorithm is that each $\mathrm{TFM}_l^{(t)}[\gamma]$ is normalized by $\mathrm{TFM}_n^{(t)}[\beta]$ to ensure $\mathrm{TFM}_n^{(t)}[\beta]=1$, where $\gamma, \beta \in \mathrm{GF}(q)$. Then, sampling a stochastic symbol is performed by comparing a uniform random number R to each $\mathrm{TFM}_l^{(t)}[\gamma]$. The output symbol is the first symbol $\gamma \in \mathrm{GF}(q)$, such that $\mathrm{TFM}_l^{(t)}[\gamma] > R$. In this case, the penalty for complicated normalization is larger area and longer latency.

In order to provide a solution with low complexity and latency, we propose a dynamic random number generation method. In the proposed decoder, the normalization of each $\mathrm{TFM}_l^{(t)}[\gamma]$ is eliminated; instead, the uniform random number R is limited to the range between 0 and $\mathrm{TFM}_n^{(t)}[\beta]$. Supposed that the bit width of TFM is M bits. First, the location of leading one of $\mathrm{TFM}_n^{(t)}[\beta]$, assuming that it is located at the kth bit and $k \leq M$, will be sorted. Then, a random number is generated by least significant $k+1$ bits of an M-bit linear-feedback shift register (LFSR). Since the random number may be larger than $\mathrm{TFM}_n^{(t)}[\beta]$, a clipping will be used for controlling the range of random numbers. Finally, the output symbol is generated by comparing the random number to each $\mathrm{TFM}_l^{(t)}[\gamma]$ as in the RHS algorithm.

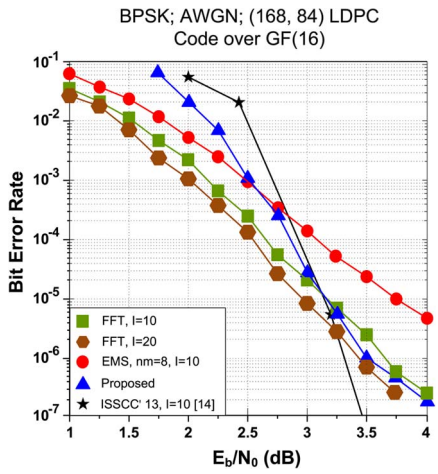

Fig. 5. BER performance of proposed decoder.

D. Decoding Process

The operations of the proposed decoder are summarized as follows.

1) *Initialization*: In the beginning, the channel values of each variable node are converted to a set of probabilities $L_v[\gamma]$ for each symbol in $\mathrm{GF}(q)$. In addition, the truncated TFM with n set of registers of each variable node are initialized by the largest n probabilities, such that $\mathrm{TFM}_l^{(t)}[\beta] = \mathrm{TFM}_{(l-1)}^{(t)}[\gamma] + L_v^2[\beta]$, where $\gamma, \beta \in \mathrm{GF}(q)$ and $l = 1, 2, \ldots, n$.
2) *Variable node update*: The VNUs follow the procedure, as shown in Fig. 3. Moreover, the overall belief is also generated by $\arg\max L_v[x]$, where $L_v[x]$ is the likelihood of the input symbol x for each variable node v [14].
3) *Symbol permutation*.
4) *Check nodes update*: The CNUs execute the finite field addition.
5) *Termination*: The decoder will be terminated whenever all of the beliefs satisfy check equations. Otherwise, the decoder generates an attempted codeword after a preset number of decoding cycles.

IV. Performance Evaluation

Based on the proposed CTFM-CC decoding algorithm and truncated TFM architecture, a (168, 84) regular-(2, 4) NB-LDPC code over GF(16) is implemented with a fully parallel architecture. To verify the functionality of the proposed decoder, computer-based software and postlayout simulations are adopted. Fig. 5 shows the BER performance of the proposed decoder. The simulation environments are binary phase-shift keying modulation and additive white Gaussian noise channel. The resolution of TFM is 11 bits, and the maximum decoding cycle is equal to 10^5 with enabled early termination. The performance curves reveal that the proposed decoder achieves the same performance compared to a floating-point FFT-SPA decoder with ten iterations $(I = 10)$ under SNR equal to 3 dB. Furthermore, it outperforms the floating-point EMS decoder when BER is below 10^{-3}. Compared to the EMS decoder in [16], the proposed decoder achieves better BER performance above 10^{-6} even if the block length of the proposed decoder is shorter than [16].

TABLE II
COMPARISON WITH THE STATE-OF-THE-ART NB-LDPC DECODERS

	This work		ISSCC' 13 [16]	TCAS-I' 12 [17]	TSP' 13 [18]	TVLSI' 13 [19]	TSP' 13 [14]
Code	(168, 84) over GF(16)		(160, 80) over GF(64)	(837, 726) over GF(32)	(837, 726) over GF(32)	(837, 726) over GF(32)	(192, 96) over GF(64)
Block length	672		960	4185	4185	4185	1152
Process (nm)	90		65	180	90	180	-
Gate count (K-gate)	1077	1253	2780	1293	8510	871	-
Frequency (MHz)	333	286	700	200	250	200	108
Latency (μs)	min: 0.51 max: 300	min: 0.60 max: 350	min: 0.69 max: 2.06	64.98	17.84	63.38	500
Throughput (Mb/s)	1318	1131	1150	64	233.53	66	698
Power (mW)	-	1174	3866	-	893	-	-
Hardware efficiency (Mb/s/K-gate)	1.220	0.903	0.413	0.049	0.027	0.076	-
Energy efficiency (nJ/bit)	-	1.038	3.362	-	3.824	-	-
Algorithm	CTFM-CC		EMS	Simplified Min-Sum	Max-Log-QSPA	Min-Max	AMSA
Status	synthesis	post-layout	measurement	synthesis	post-layout	synthesis	FPGA

The proposed decoder is implemented in a UMC 90-nm CMOS technology. Based on the synthesis result of Cadence SOC Encounter, the total gate count of proposed decoder is 1253 K. Since the decoding cycle under early termination mode is 170, the data rate of the proposed decoder achieves 1.13 Gb/s at an SNR of 3.5 dB. Table II summarizes the implementation result of the proposed decoder. To the best of our knowledge, the proposed decoder is the first over Gb/s stochastic NB-LDPC decoder.

A comparison with state-of-the-art NB-LDPC decoders is given in Table II. In [14], an FPGA-based stochastic NB-LDPC decoder using the AMSA algorithm was presented. The authors implemented a (192, 96) over GF(64) NB-LDPC decoder, where it achieved a throughput of 698 Mb/s using custom memory. In [18], a trellis-based max-log-QSPA decoder using bidirectional forward–backward recursion for a (837, 726) over GF(32) NB-LDPC code was implemented. It showed a result of a throughput of 33.53 Mb/s with 8.51 M gate count under the postlayout simulations. In [19], a relaxed check node processing scheme for the min-max NB-LDPC decoding algorithm was proposed. Based on its synthesis results, a throughput of 66 Mb/s with 871 K gate count (837, 726) NB-LDPC decoder is provided. Recently, the first silicon-proven NB-LDPC decoder was reported in [16]. A fully parallel EMS decoder for a (160, 80) over GF(64) NB-LDPC achieved a throughput of 1.15 Gb/s with 2.78 M gate count. In summary, our proposed decoder can provide a higher throughput with both better hardware and energy efficiency among related NB-LDPC decoders.

V. CONCLUSION

In this brief, an area-efficient stochastic decoder for NB-LDPC codes has been presented. The CTFM-CC decoding algorithm has been proposed to simplify operations of the variable node, and a truncated TFM architecture has been provided to reduce the complexity of VNUs under negligible performance degradation. The dynamic random number generation method has been also proposed to generate the output symbol of the SPA-to-stochastic conversion of VNU. Hence, the throughput and hardware efficiency of the decoder are enhanced. A (168, 84) over GF(16) NB-LDPC decoder for wireless communication has been implemented to demonstrate the feasibility of our proposal. The proposed decoder achieves a throughput of 1.13 Gb/s, leading to a hardware efficiency of 0.90 Mb/s/K-gate. Compared to other state-of-the-art works, our proposed design can achieve the highest hardware efficiency among available NB-LDPC decoders.

REFERENCES

[1] M. Davey and D. Mackay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[2] X. Jiang, Y. Yan, X.-G. Xia, and M. H. Lee, "Application of nonbinary LDPC codes based on Euclidean geometries to MIMO systems," in *Proc. Int. Conf. WCSP*, 2009, pp. 1–5.

[3] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[4] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[5] V. Savin, "Min-Max decoding for non binary LDPC codes," in *Proc. IEEE ISIT*, 2008, pp. 960–964.

[6] X. Zhang and F. Cai, "Reduced-complexity decoder architecture for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 7, pp. 1229–1238, Jul. 2011.

[7] Y. L. Ueng *et al.*, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 385–398, Feb. 2012.

[8] X. Chen, S. Lin, and V. Akella, "Efficient configurable decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 188–197, Jan. 2012.

[9] F. Garcia-Herrero, M. J. Canet, and J. Valls, "Nonbinary LDPC decoder based on simplified enhanced generalized bit-flipping algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 6, pp. 1455–1459, Jun. 2014.

[10] J. Lin and Z. Yan, "An efficient fully parallel decoder architecture for nonbinary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2649–2660, Dec. 2014.

[11] S. S. Tehrani *et al.*, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4883–4897, Sep. 2010.

[12] A. Naderi, S. Mannor, M. Sawan, and W. J. Gross, "Delayed stochastic decoding of LDPC codes," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5617–5626, Nov. 2011.

[13] G. Sarkis, S. Hemati, S. Mannor, and W. J. Gross, "Stochastic decoding of LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 939–950, Mar. 2013.

[14] A. Ciobanu, S. Hemati, and W. J. Gross, "Adaptive multiset stochastic decoding of non-binary LDPC codes," *IEEE Trans. Signal Process.*, vol. 61, no. 16, pp. 4100–4113, Aug. 2013.

[15] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1994.

[16] Y. S. Park, Y. Tao, and Z. Zhang, "A 1.15 Gb/s fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating," in *Proc. IEEE ISSCC*, 2013, pp. 422–423.

[17] X. Chen and C. L. Wang, "High-throughput efficient non-binary LDPC decoder based on the simplified min-sum algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2784–2794, Nov. 2012.

[18] Y. L. Ueng, K. H. Liao, C. H. Chou, and C. J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using Max-Log-QSPA," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Nov. 2013.

[19] F. Cai and X. Zhang, "Relaxed min-max decoder architectures for non-binary low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 11, pp. 2010–2023, Nov. 2013.