Rate-Compatible and High-Throughput Architecture Designs for Encoding LDPC Codes

Nishil Talati
Andrew & Erna Viterbi Faculty of Electrical Engineering
Technion - Israel Institute of Technology
Email: nishil.t@campus.technion.ac.il

Zhiying Wang
Center for Pervasive Communications and Computing
University of California, Irvine
Email: zhiying@uci.edu

Shahar Kvatinsky
Andrew & Erna Viterbi Faculty of Electrical Engineering
Technion - Israel Institute of Technology
Email: shahar@ee.technion.ac.il

***Abstract*—Low-density parity-check (LDPC) codes are known for superior performance over a wide range of codes for communication and memory systems. In many practical scenarios, adaptive ECC system is preferred that can adapt to various codes with varying channel conditions since the behavior of errors changes with time and space. This paper presents two architectural designs for efficient encoding of LDPC codes to support different code rates and lengths, which can be used for several applications. The proposed designs allow switching among different codes without any hardware modification. The first proposed design achieves extremely high throughput by removing the memory from the encoder, while still being able to adapt to a few predefined codes. The other architecture can adapt to any arbitrary code by using the memory for configuration, and yet, it achieves up to 12.9× throughput and 17.5× area improvement as compared to fully-reconfigurable encoders proposed in literature.**

***Keywords*—ECC, LDPC, NAND flash, encoder, IRA-LDPC, QC-LDPC.**

I. Introduction

Low-density parity-check (LDPC) codes [1] are a class of linear block codes that provide near-capacity performance on a large set of data-transmission and data-storage channels. Having been ignored for more than three decades due to their high complexity of encoding and decoding, these codes were resurrected in the mid-90s [2], [3], and some of the LDPC codes were shown to approach the capacity of Binary Memoryless Symmetric (BMS) channels [4]. With successful attempts to design reduced complexity decoding LDPC algorithms and their efficient circuit implementations, LDPC codes have already found their place in many commercial standards for communication systems (IEEE 802.16e, CDMA, DVB-S2), and in memory systems (flash memory).

In these standards, several codes are defined with different code rates and block lengths. An ideal LDPC system (both encoder and decoder) should be adaptive, to various code lengths and rates, according to the channel conditions. Different architectures have been proposed for variable rate, high speed LDPC decoder designs [5], [6]. However, there is still a room for improvement in performance and area efficiency in these designs.

In this paper, we propose two variable-rate encoder architecture designs, targeted to specific applications and offering unique advantages. First, a partially-reconfigurable encoder architecture is presented, suitable for ECC in multi-level cell (MLC) NAND flash memory. This architecture offers high throughput due to the absence of a memory for reconfiguring the parity-check matrix (PCM). It is, however, limited to a few predefined codes. The second proposed architecture is fully-reconfigurable and provides lower throughput than the former design, with the advantage of adapting to any code within the predefined maximum code size. Both designs are targeted to perform over binary-communication channels.

The rest of the paper is organized as follows. In Section II, we present the motivation for variable-rate LDPC system design for ECC in memory and communication systems. In Section III, we describe the concept and the proposed architectures of encoding of a special type of LDPC code. Comparison with previously proposed encoder designs is presented in Section IV, and Section V concludes this paper.

II. Motivation

A wireless communication system, where it is possible to decide the code rate before transmission according to the SNR of the channel, and a memory system, where the bit-error-rate increases with lifetime of the memory, are the two targeted applications of the proposed designs. A system with ability to adapt different codes gains higher error correcting performance in order to support different sensitivities of communication channels and to enhance the lifetime of memory systems. In this section, we discuss detailed motivation to consider reconfigurable ECC designs with case studies of MLC NAND flash memory and wireless communication channels.

A. Memory Systems

In a typical memory system, the rate of error depends on several factors, and varies with respect to time and space. For example, for NAND flash memory, as the memory usage increases, the memory cells start wearing out, and, the number of errors increases. Therefore, the necessity for stronger ECC increases in the latter life of the memory. Furthermore, in MLC NAND flash memory, different bits within each memory cell belong to different pages. As observed in [7], the upper pages are subject to higher bit error rates than the lower pages, due to the nature of intercell bit mapping. Hence, employing an equal code redundancy might result in either degradation in the error correction performance in the upper page or unnecessary read latency in the lower page.

B. Time-Varying Wireless Communication Channels

A wireless communication signal traveling in an environment full of scattered objects is reflected and deflected, which results in multipath transmission. This multi-path transmission adds unnecessary delay and phase to the signal. Furthermore, the changes in the environment due to movement of the

objects introduce time-varying fading on the communication channel. The rates of change of amplitude and phase of the transmitted signal deviate from the original signal depending on the velocity of objects, transmitter, and receiver. Hence, rate compatibility is an important design challenge in a system operating over time-varying channels [8], [9].

III. Proposed Reconfigurable Encoders

Motivated by examples shown in the previous section, we propose two different low-complexity reconfigurable encoder architectures. First, we explain the concept of encoding of a special type of LDPC code called Irregular Repeat Accumulate (IRA). Then, an encoder architecture that supports a few pre-defined Parity-Check Matrices (PCMs) is presented, followed by a fully-programmable encoder architecture that supports any PCM within a particular maximum code-word size. Note that discarding some of the parity bits while transmission, a method known as puncturing [10], can be used to obtain rate-compatible systems. However, a puncture-based system requires a depuncturing mechanism, and transmission at a higher rate suffers from performance degradation [9]. Hence, in this paper, we invest efforts in designing a variable-rate system without puncturing, which does not require any additional hardware for depuncturing and maintains the performance even at higher rates. Furthermore, we choose IRA-LDPC codes due to the regular structure of its PCM and the simpliciy of encoding, other types of LDPC codes such as Quasi-Cyclic (QC)-LDPC codes which have circulant form of PCM are left for future work.

A. Encoding of IRA-LDPC Codes

Irregular Repeat Accumulate (IRA) LDPC codes [11] are of interest because of their simple encoder structure and close-to-capacity performance. In this paper, we consider systematic IRA codes, where the codeword is of the type $(u_1, u_2, ..., u_k; p_1, p_2, ..., p_{n-k})$. Here, u's represent k information bits, and p's represent $(n-k)$ parity bits. The PCM of IRA codes has the form of

$$H = [H_u \; H_p], \tag{1}$$

where H_p is an $(n-k) \times (n-k)$ bi-diagonal/dual-diagonal matrix, which means that $h_{i,j} = 1, \forall i = j, j+1$. Furthermore, H_u is an $(n-k) \times k$ matrix with column and row weights (w_c, w_r), which do not grow with the size of PCM.

Most of the well-known LDPC encoder architectures [12]–[14] compute the parity bits by using the generator matrix G, where the complexity of encoding is up to $\mathcal{O}(n^2)$ for the whole parity?. We solve the encoding problem using PCM, where the time complexity of encoders for the proposed architectures is as low as $\mathcal{O}(1)$ for the whole parity. In our architectures, the values of parity bits are determined by the constraint equations from the PCM. For example, the value of j^{th} parity is determined by

$$p_j = p_{j-1} \oplus \sum_{i=1}^{k} h_{i,j} \cdot u_i, \tag{2}$$

where $h_{i,j}$ is the $(i,j)^{th}$ entry in H, and $p_0 = 0$. In order to encode the information bits with a fixed H, a *non-reconfigurable* architecture similar to [15] could be used, which only contains a series of XOR gates, and the connections from the stream of information bits to the XOR gates is hard-wired. The time complexity of this architecture is $\mathcal{O}(1)$. However, it is not suitable for communication channels where rate-compatibility of ECC is required. Design of low-complexity, yet reconfigurable ECC systems is a non-trivial problem, and in further subsections, we propose the LDPC encoder architectures for such systems to execute (1) and (2).

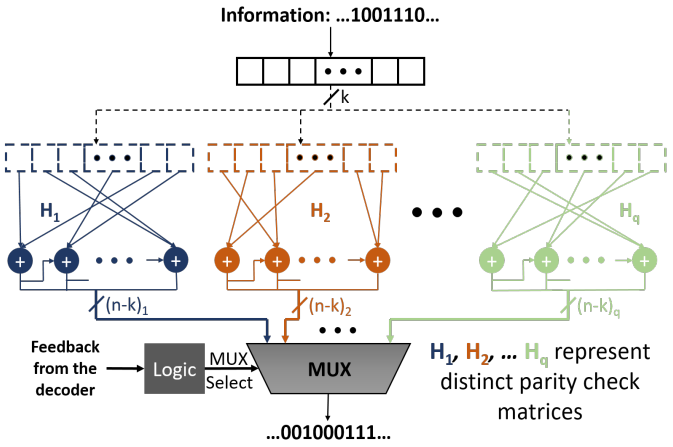

Fig. 1. Architecture of the generic partially-reconfigurable encoder, which can support q predefined PCMs $\{H_1, H_2, ..., H_q\}$. Each color corresponds to a distinct connection pattern pertaining to a particular PCM, when a single pattern is selected by the multiplexer.

B. Partially-Reconfigurable Encoder Architecture

To attack the rate-compatibility problem while maintaining performance equivalent to that of non-reconfigurable architectures, we propose to integrate q-hard-wired encoders for the set of q PCMs $\{H_1, H_2, ..., H_q\}$, and select the output of one of them using a multiplexer, as shown in Fig. 1. The feedback from the decoder regarding error rates is used to encode the select lines of the MUX, using a logical block (for example, analog-to-digital converter), and thus, the appropriate code is selected according to the channel conditions among q distinct possibilities.

While this architecture seems relatively simple, it is well-suited for the MLC NAND flash and similar systems. Because the PCM is stored in an SRAM, the necessity to configure and access it during runtime is the major bottleneck in terms of performance and area efficiency in reconfigurable architectures. This architecture achieves extremely high performance by removing SRAM from the architecture (as opposed to [12], [14]). For example, while encoding data-stream in different pages for MLC NAND flash memory in continuum, encoding a portion of the stream using a matrix stored in SRAM within the encoder for the one page, and then, configuring the matrix and encoding the other portion of the stream for the other page is not an efficient solution. In such cases, the proposed architecture can be employed to encode different portions of the data-stream with different rates. Doing so yields significant performance improvement as compared to fully-reconfigurable encoders [12]–[14]. Furthermore, for memory systems where the number of errors increases with the age of the memory (for example with retention and endurance failure of cells), this design with $q > 2$ can be used to dedicate different H_i's for different ages of memory. The only limitation of this

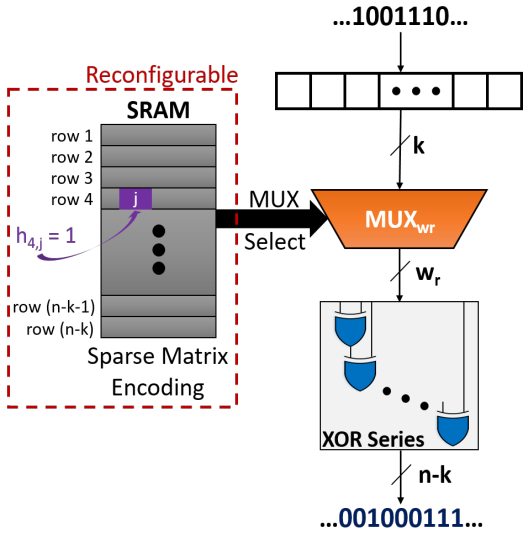

Fig. 2. Architecture of the fully-reconfigurable encoder with $(n-k)$ parity bits. The PCM is stored by sparse encoding, *i.e.,* by storing the locations of 1's from each row of H in the SRAM.

architecture is its inability to adapt to any arbitrary code, which can be mitigated to a large extent by using a large enough value of q.

C. Fully-Reconfigurable Encoder Architecture

To remove the limitation of the partially-reconfigurable architecture, we propose a fully-reconfigurable architecture to adapt to any arbitrary code rate by configuring the matrix stored in SRAM. We design the architecture to minimize the size of the SRAM required to store the PCM and use it to encode the incoming data-stream, as shown in Fig. 2.

In this architecture, the sparse property of the PCM is exploited, and before storing H in SRAM, the sparse encoding is carried out. Thus, the SRAM only contains the matrix H_u in sparse form, *i.e.,* the location of 1's in each row is stored instead of the full matrix. We do not store the rest of the matrix, *i.e.,* H_p, since it always follows a regular form. This design reduces the memory requirement of the encoder tremendously, and is especially beneficial for very lengthy LDPC codes of large code length (for example, DVB-S2).

It can be concluded from (2) that the XOR operations for any p_i are performed only among bits of the set $S_1 = \{u_i |\ h_{i,j} = 1\}$ and p_{i-1}, and the rest of the bits from the set $S_2 = \{u_{i'} |\ h_{i',j'} = 0\}$ can be ignored since $h_{i',j'} \cdot u_{i'} = 0$. This fact can be exploited to further simplify the hardware of the encoder. A series of multiplexers is used to select the bits from the information vector that belong to S_1, and the rest of the bits are ignored. For H_u with row weight w_r, the number of multiplexers required is w_r. Finally, to find the parity bits, these w_r outputs of the multiplexers are inputted into a series of XOR gates, along with the previously generated parity bit (assuming each XOR gate is 2-input, the total number of XOR gates required is w_r). In this architecture, each parity bit is generated in one clock cycle, and the clock period is determined by the delays in SRAM access, by the multiplexer, and by a series of XOR gates. The complexity of encoding is $\mathcal{O}(n-k)$.

The reconfiguration of the ECC code in this architecture can be carried out either by keeping k constant or by keeping $(n-k)$ constant. In the former case, the number of encoding cycles would vary, depending on the code rate, and in the latter case, the number of encoding cycles would be constant. The initial size of the SRAM is set according to the maximum possible size of H_u in the sparse form. Furthermore, the number of multiplexers and XOR gates are determined by the maximum row weight of the prospective H_u. Thus, any matrix having fewer hardware requirements than the designed hardware can be used for encoding the information stream. This architecture is appropriate for applications where the PCM is not changed frequently (since it incurs the sparse encoding complexity), and the rates are required to be set arbitrarily.

IV. Experimental Results and Discussion

We compare the two proposed architectures with previously proposed LDPC encoder architectures [12]–[15]. We compare the degree of reconfiguration, the SRAM size required to store PCM, the number of encoding cycles, the total area (in terms of two-input NAND gates), and the throughput of the encoders. In order to calculate performance and area, all the circuit components in the proposed architecture, including SRAM and 512-to-1 MUX, have been synthesized. All the mentioned architectures are designed in Verilog HDL and synthesized at 28nm using Synopsys.

The size of the information vector is fixed at $k = 512$, and the code rate is varied in the interval $[0.5, 0.99]$ in fully-reconfigurable architectures. For quasi-cyclic (QC)-LDPC encoders [12], [13], the parameters $(M, N, b)_{max}$ are set to $(4, 8, 128)$, where b represents the order of the circulant permutation matrices in the generator matrix. In this comparison, for all the fully-reconfigurable architectures, the configuration of ECC code is carried out by keeping the value of k constant, and $(n-k)$ is varied with code rate r. Furthermore, for IRA-LDPC codes, we choose $(5, w_r)$-code; in other words, PCMs are generated using $w_c = 5$, since the error-correction performance at this w_c is extremely high with significantly low error floor [16]. For the proposed partially-reconfigurable architecture, $q = 4$ and the rates are set to $\{0.88, 0.92, 0.96, 0.98\}$, which is appropriate for a memory application. Table I shows a comparison of different encoders at $r = 0.9$.

As expected, the non-reconfigurable architecture [15] achieves the highest throughput due to hard-wired connections and lack of memory access. However, this architecture cannot support systems that require variable code rate. Among reconfigurable architectures, the partially-reconfigurable architecture achieves the highest throughput since the encoding path does not include any memory access.

Among the fully-reconfigurable architectures, *i.e.,* [12], [14], and this work, it can be concluded that QC-LDPC codes achieve better area efficiency in terms of matrix storage because of their regular structure. However, these codes increase the number of encoding cycles since they shift, in every cycle, the contents of first rows in the circulant permutation matrices [12]. The proposed fully-reconfigurable encoder is $17.5\times$ more area-efficient than [14] in terms of matrix storage efficiency. Furthermore, the proposed architecture achieves the highest throughput, with a 22.6% increase in area as compared to [12], mainly because of the SRAM area.

TABLE I. COMPARISON OF DIFFERENT ARCHITECTURES FOR MAXIMUM PCM SIZE 512×1024, $k = 512$, AND $r = 0.9$

	[13]	[15]	Proposed Part.-Recon. Architecture (q = 4)**	[12]	[14]	Proposed Fully-Recon. Architecture
Type of Code	QC-LDPC	EG-LDPC	IRA-LDPC	QC-LDPC	IRA-LDPC	IRA-LDPC
Reconfiguration	Not Possible	Not Possible	Partially	Fully	Fully	Fully
SRAM Size	$4 \times (8 \times 128)^{+*}$	–	–	$16 \times 128^{+}$	512×1024	512×64
#Encoding Cycles	512	1	1	64	57	57
Area (per NAND2)	$75.6k$	$0.09k$	$2.5k$	$45.3k$	$1030.9k$	$58.5k$
Throughput (Gbps)	0.487	836.8	606.25	2.78	0.277	3.57

[+]The selected parameters of QC-LDPC code are: $(M, N, b)_{max} = (4, 8, 128)$.
[*]This architecture requires four private ROMs of size 8×128.
[**]Four different rates chosen are: $r = \{0.88, 0.92, 0.96, 0.98\}$.

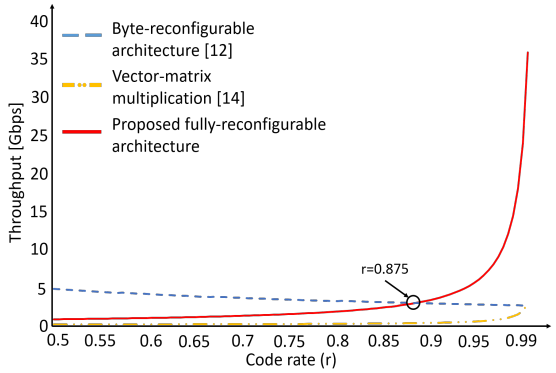

Fig. 3. Throughput of different fully-reconfigurable encoder architectures for any arbitrary code rates from 0.5 to 0.99. The comparison shows that the proposed encoder achieves the highest throughput for $r > 0.875$.

For performance, we compare the throughput of the fully-reconfigurable architectures for different code rates in Fig. 3. The proposed architecture achieves on average $12.9\times$ higher throughput than [14]. The throughput of the proposed encoder is the highest among all architectures for $r > 0.875$, and for $r < 0.875$, it is inferior to [12] because of the fixed number of encoding cycles that is equal to the number parity bits.

V. CONCLUSION

Some memory and communication systems require supporting multiple codes with different code rates and code lengths. In this paper, we propose two efficient encoding hardware designs for reconfigurable IRA-LDPC codes that can be used for such systems. One of the proposed architectures is suitable for applications where the transition among different codes is required quite frequently, and the number of possible codes is finite. This partially-reconfigurable architecture achieves significantly higher throughput than the fully reconfigurable encoder designs. The second proposed architecture exploits the sparse property of the PCM, and reduces the hardware requirement by storing the matrix in SRAM in its sparse form. This sparse form is then used to encode the input data stream, and it performs faster than previously proposed encoder architectures. This architecture is appropriate for applications where the code is changed less frequently, and adaptation to any arbitrary code is required.

ACKNOWLEDGMENT

This research is partially funded by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI), and by the Viterbi Fellowship in the Technion Computer Engineering Center, and partially by NSF grant CCF-1566587. The authors would like to thank Eric Herbelin and Goel Samuel from the Technion for their excellent technical support.

REFERENCES

[1] R. G. Gallager, "Low-density Parity-Check Codes," 1963.

[2] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," in *IEEE Trans. on Information Theory*, 1996, pp. 1723–1731.

[3] D. J. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[4] S. Kudekar, T. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. on Information Theory*, vol. 59, no. 12, pp. 7761–7813, 2013.

[5] G. Masera, F. Quaglio, and F. Vacca, "Implementation of a flexible LDPC decoder," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 54, no. 6, pp. 542–546, June 2007.

[6] J. Y. Lee and H. J. Ryu, "A 1-Gb/s flexible LDPC decoder supporting multiple code rates and block lengths," *IEEE Trans. on Consumer Electronics*, vol. 54, no. 2, pp. 417–424, May 2008.

[7] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit error rate in NAND flash memories," in *2008 IEEE International Reliability Physics Symposium*, April 2008, pp. 9–19.

[8] O. Jetlund, G. Oien, K. Hole, V. Markhus, and B. Myhre, "Rate-adaptive coding and modulation with LDPC component codes," *European Cooperation in the Field of Scientific and Technical Research*, Sept. 2002.

[9] C. P. Fewer, M. F. Flanagan, and A. D. Fagan, "A versatile variable rate LDPC codec architecture," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 54, no. 10, pp. 2240–2251, Oct 2007.

[10] D. G. M. Mitchell, M. Lentmaier, A. E. Pusane, and D. J. Costello, "Randomly punctured LDPC codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 408–421, Feb 2016.

[11] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, 2000, pp. 1–8.

[12] Y. M. Lin, H. T. Li, M. H. Chung, and A. Y. Wu, "Byte-reconfigurable LDPC codec design with application to high-performance ECC of NAND flash memory systems," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 62, no. 7, pp. 1794–1804, July 2015.

[13] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. on Communications*, vol. 54, no. 1, pp. 71–81, Jan 2006.

[14] H. Yasotharan and A. C. Carusone, "A flexible hardware encoder for systematic low-density parity-check codes," in *2009 52nd IEEE International Midwest Symposium on Circuits and Systems*, Aug 2009, pp. 54–57.

[15] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for NanoMemory applications," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 4, pp. 473–486, April 2009.

[16] Y. Zhang and W. E. Ryan, "Structured IRA codes: Performance analysis and construction," *IEEE Trans. on Communications*, vol. 55, no. 5, pp. 837–844, May 2007.