A (50,2,4) Nonbinary LDPC Convolutional Code Decoder Chip over GF(256) in 90nm CMOS

Chia-Lung Lin, Chih-Lung Chen, Hsie-Chia Chang, and Chen-Yi Lee
Department of Electronics Engineering & Institute of Electronics National Chiao Tung University,
1001 Ta-Hsueh Road, Hsinchu, Taiwan, R.O.C.
Tel: +886-3-5712121 ext: 54238 email: {jllin, lung, hcchang, cylee}@si2lab.org

***Abstract*—A memory-based ($m_s = 50$,$d_v = 2$,$d_c = 4$) nonbinary LDPC convolutional code (NB-LDPC-CC) decoder over GF(256) with layered scheduling is presented. The proposed architecture-aware construction features fewer memory banks, low degree, low period, and better performance. To the best of our knowledge, this is the first architecture discussion and implementation for NB-LDPC-CC decoders. We optimized the architecture of message first-in-first-out (M-FIFO), check node unit, and variable node unit in terms of area and throughput. Jointly designing code and architecture, overall normalized area efficiency can be enhanced by more then six times with respect to decoders of nonbinary LDPC block codes (NB-LDPC BCs). After fabricated in 90nm CMOS, our prototype NB-LDPC-CC decoder chip can achieve maximum throughput of 22.8Mbps with frequency of 285MHz. The measured average power is 211mW at a typical operating voltage of 1.0V.**

I. Introduction

Error-correcting performance of binary LDPC codes is degraded when the code size is short, or under a higher order modulation. In [1], the authors showed that the error-correcting capability of binary LDPC codes can be greatly enhanced by a generalization to higher order Galois fields, GF(2^p) with $p > 1$. However, a major drawback of these codes is their decoding complexity, increasing with the order of the field, which discourages implementations of these codes from high values of finite field size. Therefore, many studies address how to decrease the complexity while maintaining high performance.

Several decoding methods of binary LDPC convolutional codes are presented. Nonbinary LDPC block codes decoding algorithms also can be applied to its convolutional counterparts. Moreover, taking the advantage of convolutional structure, the distance between two variables checked by one check-sum node is limited by the maximum degree of generator polynomials, the sliding window decoding [2] and pepeleine decoder [3] architecture decoder are more efficient.

In this paper, we start from construction to design a architecture-aware code. The proposed code result in decoder fewer memory banks, more simple processing unit (low degree), while maintain good performance. In the decoder architecture, we proposed modified trellis-based min-max with layered scheduling, and message FIFO structure. By all above effort, the synthesis and measurement results are shown to prove high area efficiency.

This work was supported by National Science Council, R.O.C., under project NSC 100-2221-E-009-044-MY3 and NSC 101-2220-E-009 -060.

This paper is organized as follows. Section II is the proposed architecture-ware construction of NB-LDPC-CCs. Section III will describe the proposed algorithm for NB-LDPC-CC architecture. Then, Section IV gives the implementation results and comparison. Finally, the conclusion is given in Section VI. Besides, this report only concerns about GF(2^p)-LDPC codes for convenience.

II. Architecture-aware NB-LDPC-CCs

In the following, we present a construction method of a (m_s,$d_v = 2$,$d_c = 4$) regular NB-LDPC-CC over GF($q = 2^p$) for phase $T > 2$, where m_s, d_v, and d_c are memory length, variable node degree, and check node degree respectively. With given m_s, $T > 2$, and GF(2^p), the constructed time-varying (m_s,2,4) regular LDPC-CCs over GF(2^p) guarantee girth > 12.

For the sake of clarity, we use syndrome former in polynomial domain to present this method. The construction method will determine the two polynomials of $h_t^{(1)}(D)$ and $h_t^{(2)}(D)$, where t is the phase index. The code construction are shown in the following:

1) Choose four non-zero elements randomly, α_1,α_2,α_3,α_4 $\in$ GF(2^p).
2) Choose two integers, m_b and m_d, satisfying the following constraints
   - $m_s/2 < m_b < m_d < m_s$
   - $m_d - m_b \bmod T(T-1) = 0$
   - $m_s = m_d \bmod T$, and $m_s \neq 0 \bmod T$
   - $(T-1)(m_s - m_d) \neq (m_d - m_b)$
3)

for($t = 1$ to T
$h_t^{(1)}(D) = \alpha_1 + \alpha_2 D^{m_s}$;
$h_t^{(2)}(D) = \alpha_3 + \alpha_4 D^{m_b+(t-1)(m_d-m_b)/T}$;
endfor

Assume the codeword is denoted as $\mathbf{Z} = [\mathbf{z_1}, \mathbf{z_2}, ..., \mathbf{z_t}, ...]$, where $\mathbf{z_t} = (z_{t,1}, z_{t,2})$. In the proposed codes, every check nodes connecting to four variable nodes. Two of them belong to first elements of $\mathbf{z_t}$, and the other two belong to the second element of $\mathbf{z_t}$. Considering implementation, following features lead cost-efficient decoder. 1) There is only one term in syndrome former is time variant, which reduces the number of memory banks. 2) Small and regular d_v and d_c are

not only decreases the complexity of check node units and variable node units considerably, but also lower the storage size. Besides, 3) $\alpha_1 \sim \alpha_4$, are time invariant, that simplifies the controller. 4) Phase is flexible. Choosing small period releases the loading of multiplexer. In terms of error-correcting capability, the girth of NB-LDPC-CC constructed by proposed method is always larger than fourteen.

The following is a constructed of (50,2,4) NB-LDPC-CC over GF(2^8) with $\alpha_1 = 123$, $\alpha_2 = 219$, $\alpha_3 = 177$, $\alpha_4 = 191$, $m_b = 29$, $m_d = 47$ and $T = 3$. The corresponding syndrome former in polynomial domain is show in eq. (1). The puncture scheme of [4] is also implied to the constructed code. The simulation results are shown in 1.

$$\begin{aligned}
\mathbf{H}_1^T(D) &= \begin{bmatrix} h_{(1)}^{(1)}(D) \\ h_{(1)}^{(2)}(D) \end{bmatrix} = \begin{bmatrix} 123 + 219D^{50} \\ 177 + 191D^{29} \end{bmatrix} \\
\mathbf{H}_2^T(D) &= \begin{bmatrix} h_{(2)}^{(1)}(D) \\ h_{(2)}^{(2)}(D) \end{bmatrix} = \begin{bmatrix} 123 + 219D^{50} \\ 177 + 191D^{38} \end{bmatrix} \\
\mathbf{H}_3^T(D) &= \begin{bmatrix} h_{(3)}^{(1)}(D) \\ h_{(3)}^{(2)}(D) \end{bmatrix} = \begin{bmatrix} 123 + 219D^{50} \\ 177 + 191D^{47} \end{bmatrix}
\end{aligned} \tag{1}$$

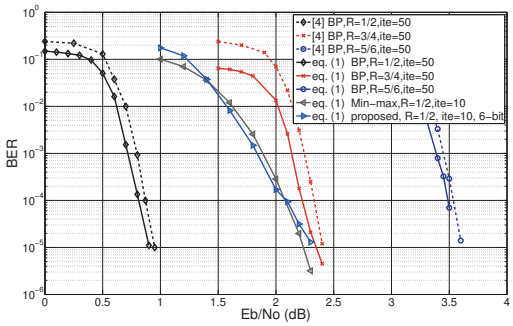


Fig. 1. Simulation results of NB-LDPC-CC listed in eq. (1) with different decoding algorithms. Curves in [4] are included for comparison.

III. Memory-based layered scheduling with modified min-max algorithm

A. Decoder architecture

Overall decoder architecture for eq. (1) is shown in Fig. 2. Define every S_c clock cycles as a elementary operation time unit, τ, which corresponds to the delay of polynomial in eq. (1). The decoder is composed of I processors, where I also equals to the number of iteration, and all processors are identical. Each processor consists of two Message-FIFOs (M-FIFO$_\mathrm{A}$ and M-FIFO$_\mathrm{B}$), a check node unit (CNU) and four reduced variable node units (R-VNUs). M-FIFO$_\mathrm{A}$ stores the data of $z_{t,1}$ and M-FIFO$_\mathrm{B}$ stores the data of $z_{t,2}$ within $(m_s + 1)\ \tau$.

Timing diagram of key modules is shown in Fig. 2. At the beginning of τ, two M-FIFOs send oldest data to next processor, receive new data from previous processor, and deliver data to CNU and R-VNUs. In last cycle of τ, copy the updated variable node data from R-VNUs to two M-FIFOs. In every τ, each processor updates a check node and then updates the d_c neighbouring variable nodes connecting to the check node. The more detail information are gave in the following.

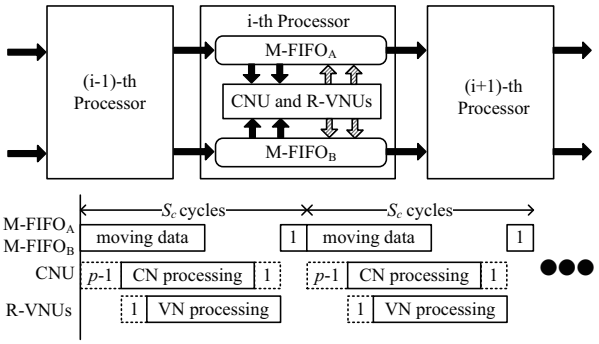


Fig. 2. Decoder architecture and timing diagram of key modules.

B. Decoding algorithm

The check node processing and variable node processing are show in eq. (2) and eq. (3) respectively. $\mu_j(\alpha)$ is the variable node message of element α, and $\nu_i(\alpha)$ is the check node message of element α. In the initialization, assign $L(\alpha) = \frac{p(r|\alpha')}{p(r|\alpha)}$ to $\mu_j(\alpha)$, where r is received data, and α' is the element in GF(2^p) which maximize prior probability of the symbol with respect to the channel. Because our target finite field size is 256, it's impossible to process all elements. Besides, in order to achieve better trade-off between performance and area cost, only the $n_m < q$ most reliable messages are kept for variable node messages and check node messages to reduce storage size and complexity of computation units. $V_i[]/U_j[]$ denotes the sorted variable/check node message vector which stores the smallest n_m values in μ_j/ν_j, and $A_j[]/B_j[]$ stores corresponding symbols. In variable node processing, using $U_j[n_m] + \gamma$ to substitute the reliabilities of the symbols which are not stored in U_j.

$$\begin{aligned}
&\text{for} \quad i = 1 \text{ to } d_c \\
&\quad \nu_i(\alpha) = \min_{\{\alpha_1 + \cdots + \alpha_{d_c} = 0 | \alpha_i = \alpha\}} \ \max_{j \in \{1, \cdots, d_c\}/i} \mu_j(\alpha_j)
\end{aligned} \tag{2}$$

$$\mu_j(\alpha) = L(\alpha) + \nu_j(\alpha) \quad \text{for} \quad j = 1 \text{ to } d_c \tag{3}$$

In [5], the trellis-based min-max algorithm has two main steps, and iteratively executes them. In first step, to find the minimum value among $U_{1 \sim d_c}[]$, and then to exclude it. In second step, path constructor determine whether the value found in first step should be included in $V_i[]$, and calculate the corresponding symbol.

C. Processing units

The block diagram of processing units and data flow are show in Fig. 3(a). There are five units in a processor, one CNU for check node processing with modified trellis-based min-max algorithm [5], and four R-VNUs for variable node processing. The proposed CNU is composed of one $sorter_A$ and four Path Constructors (PCs). Considering ASIC implementation, we jointly designed M-FIFOs and CNU to delete some memory banks and further avoided worst case to improve performance. Moreover, the proposed R-VNU decrease the latency and storage for channel likelihood.

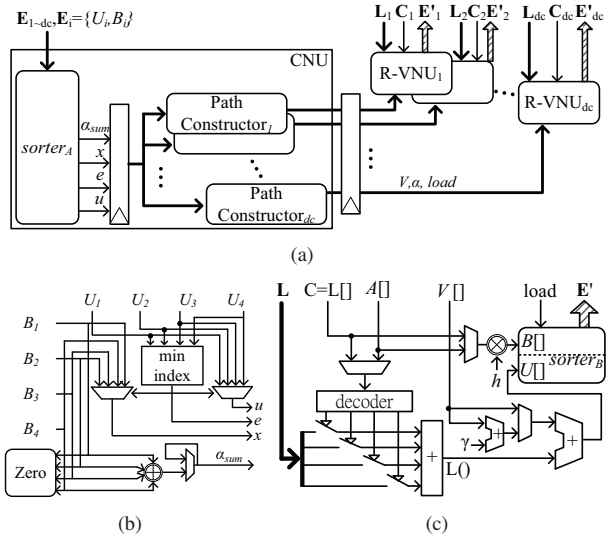


Fig. 3. (a) CNU and R-VNUs (b) $sorter_A$ with $d_c = 4$ (c) R-VNU over GF(2^4)

The architecture of $sorter_A$ used in CNU with $d_c = 4$ is described in Fig. 3(b), where u is the output reliability, e is the output degree index, x is the output symbol, $Zero$ and α_{sum} are the same function in [5]. We further define $U_1 \sim U_4$ and $\beta_1 \sim \beta_4$ as $\mathbf{E_{1\sim d_c}}$. Jointly designing two M-FIFOs and CNU, the variable node message buffer is eliminated. In other words, $sorter_A$ accesses variable node messages from the two-FIFOs directly.

The path constructor is similar to the one in [5]. We use some negligible control circuits to improve worst case. When the most reliable values are all in one variable node, the worst case occur. The case result in path constructor works very inefficiently. Furthermore, the worst case will be worse quadratically as n_m increasing linearly. In order to release the case, we added some control circuits to detect the worst case, and speed up path constructors.

Fig. 3(c) is the architecture of proposed R-VNU over GF(2^4). According to eq. (3), VNU needs to calculate out $\mu_j(\alpha)$ for all symbols, and then sorts out the n_m smallest values. In conventional variable node processing [6], variable node processing is implemented by forward/backward algorithm, taking $2n_m$ cycles to finish an elementary variable node unit. Taking the advantages of proposed code, $d_v = 2$, the VNU can be implemented by on elementary unit with layered scheduling. Besides, we further reduce the latency and storage for channel symbols from $2n_m$ to $(n_c + 1) < 2n_m$, where n_c is the number of considering channel symbols. We named the VNU as reduced VNU (R-VNU). The architecture over GF(2^4) and corresponding equation is show in Fig. 3(c) and eq. (4) respectively, where $sorter_B$ in 3(c) is the same as the sorter in [6], and $\mathbf{L}$ in 3(c) means the channel values of p bits in log likelihood ratio. In our implementation, $n_c = 8$ with about 0.2dB performance lose.

$$U_j[0 \sim n_m - 1] = sort \left\{ \begin{array}{l} V_j[y] + L_j(A_j[y]), 0 \le y < n_m \\ L_j[x] + V_j[n_m - 1] + \gamma, 0 \le x < n_c \end{array} \right\} \quad (4)$$

D. M-FIFOs

Fig. 4 depicts the architecture of M-FIFO$_{\mathrm{A}}$ and M-FIFO$_{\mathrm{B}}$. Each M-FIFO stores three different kinds data of variable nodes within (n_m+1) τ. L-MEM stores channel values of each the bits; C-MEM stores the compensation symbols for variable node processing ($L[]$); E-MEM stores variable node messages. Applying layered scheduling, the decoder only saves one degree variable node messages, which is connecting to next time of check node processing. Except L,C,E-MEM$_{*2}$ are two ports memories, which support read and write for different addresses simultaneous, the other six memories are single port memories. In our implementation, we choose $n_c = 8$, so we can combine every MEM$_{L*}$ and MEM$_{L*}$ to one memory bank. Therefore, the number of memory banks per processor is six. The numbers in the figure are the delay indexes of stored variable nodes, and apostrophe means the buffers for updated variable node. The storages above E-MEM$_{B2}$ are input and output buffers of the variable node data for CNU, which are in the L,C,E-MEM$_{*2}$ and are needed by CNU.

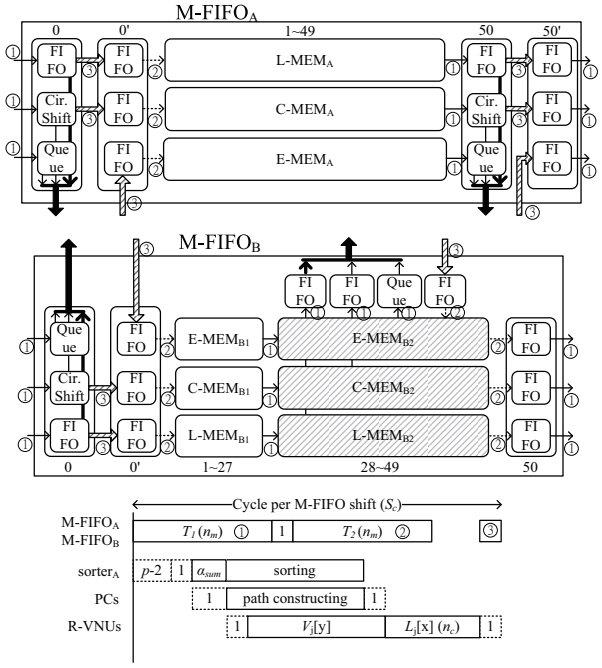


Fig. 4. Timing diagram of key modules, and architecture of M-FIFO$_A$ and M-FIFO$_B$

The timing diagram of M-FIFO$_A$, M-FIFO$_B$, and other processing units is in the bottom of Fig. 4. In a elementary operating time unit, the data flow can be divided to three phases, T_1, T_2, and last cycle, and the corresponding data flow are the matched number, ②, ③, and ④. The timing diagram of $sorter_A$, PCs and R-VNU are also shown in the timing diagram.

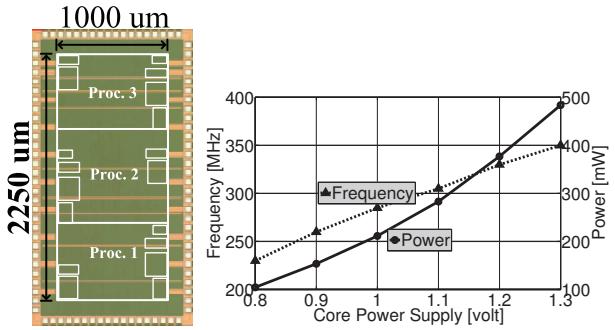
IV. Implementation and Comparison

A. Chip implementation

The proposed (50,2,4) LDPC-CC decoder over GF(256) with $n_m = 32$, including channel sorter [7] and decision unit,

fabricated in 90-nm 1P9M CMOS technology. The quantization is 6 bits, and S_c is configurable. Because of the area limitation, the decoder only implements three processors. The die photo is shown in Fig. 5(a) with the processors distribution, where the dash blocks denotes the memory banks. The core area is 2.25 mm$\times$ 1.00 mm with gate counts of 524K (654K if including memory area). The performance of proposed architecture with $I = 10$ is shown in Fig. 1.

The number of memory banks is 3*6=18, and over all memory size is 166.3 Kb. Under SNR = 2.0 dB, the measurements show that a clock frequency of 285 MHz can be achieved, indicating a maximum throughput of 22.8 Mbps under 1V. Fig. 5(b) shows the measurement results of power and frequency under different core power supply when $S_c = 200$ and $SNR = 2.0$ dB. In the full-speed case, the clock frequency of 350 MHz can be achieved at 1.3V and the power dissipate 483.6 mW with throughput of 28 Mbps. To achieve low-power purpose, the core supply voltage can be scaled to 0.8 V with power consumption of 104 mW under frequency 230 MHz.



(a) (b)

Fig. 5. (a) Die photo (b) Measurement results of power and frequency

B. Comparison

Chip specification and implementation results is summarized in TABLE I, where the state-of-the-art are listed for comparison. The number of gate count listed in this table includes memory area, and the gate count of [5] is estimated in [8]. The implementation results or this work is under SNR = 2.0dB, $S_c = 200$. We also list synthesis results of 10 processors (iteration) for comparison.

For fairly comparing area efficiency, performance normalization is required. The throughput of all designs are almost linear to n_m. Assume the normalization factor of n_m for gate count is n_m, too. Gate count of NB-LDPC-CC decoders are linear to iteration, and throughput of NB-LDPC decoders are linear to iteration inverse. The (area) efficiency equation is show in eq. (5). TABLE I shows that the proposed architecture is more then six times $(171.8/24, 107/15.8)$ more efficient then previous designs.

$$\text{Efficiency} = (\text{iteration} * \text{n}_\text{m}^2 * \text{Mbps/Kgates}) \quad (5)$$

V. Conclusion

We have presented a memory-based nonbinary LDPC convolutional code (NB-LDPC-CC) decoder over GF(256) with layered scheduling. To improve the area efficiency, we start from designing code construction. The proposed codes feature regular, low degree, low period. Moreover, in terms of performance, the constructed code outperform then previous work. Taking the advantages of our code, we proposed efficient message FIFO structure. For throughput enhancement, we design R-VNU to shorten VNU latency, modified min-max algorithm with layered scheduling, and proposed path constructor which avoids worst case of latency. For area efficiency, we reduce the storage for check node message, variable message, and channel information. Fabricated in 90nm CMOS process, the chip occupies 2.25 mm^2 area and achieves maximum throughput 22.8 Mbps can be achieved at 1V. Comparing to the state-of-the art of NB-LDPC decoders, area efficiency of our prototype NB-LDPC-CC decoder can be enhanced by more then six times.

TABLE I
Comparison of previous designs

	This work (synthesis)	This work (measured)	[8] post-layout	[5] (synthesis)
Code length	816-bit (constraint length)		1240-bit	4185-bit
Field/n_m	GF(256)/32		GF(32)/8	GF(32)/16
Code type	NB-LDPC-CC with $T = 3$, $n_m = 50$		NB-LDPC-BC	
Code rate	0.5/0.75/0.875		0.55	0.87
Quantization	6 bits		7 bits	5 bits
Process	90nm		90nm	N/A
Iteration	10	3	10	15
Frequency	430 MHz	285 MHz	260 MHz	150 MHz
Throughput	34.4 Mbps	22.8 Mbps	47.69 Mbps	10 Mbps
Gate count	205*10 K	654 K	1,920 K	1,600 K
Core area	N/A	2.25 mm^2	10.33 mm^2	N/A
Power	N/A	211mW	N/A	N/A
Efficiency	171.8	107	15.8	24.0

References

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

[2] A. J. Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity check matrix," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[3] A. E. Pusane, A. J. Feltstrom, A. Sridharan, M. Lentmaier, K. Zigangirov, and D. J. Costello, "Implementation aspects of LDPC convolutional codes," *IEEE Trans. Commun.*, vol. 56, pp. 1060–1069, 2008.

[4] K. K. H. Uchikawa and K. Sakaniwa, "Terminated LDPC convolutional codes over GF(2p)," in *Annual Allerton Conference on Communication, Control, and Computing*, 2010, pp. 195–200.

[5] X. Zhang and F. Cai, "Reduced-complexity decoder architecture for non-binary LDPC codes," *IEEE Trans. VLSI Syst.*, vol. 19, no. 7, pp. 1229–1238, 2011.

[6] A. Voicila, F. Verdier, D. Declercq, M. Fossorier, and P. Urard, "Architecture of a low-complexity non-binary LDPC decoder for high order fields," in *International Symposium on Communications and Information Technologies (ISCIT)*, 2007, pp. 1201–1206.

[7] A. A. Ghouwayel and E. Boutillon, "A systolic LLR generation architecture for non-binary LDPC decoders," *IEEE Communications Letters*, vol. 15, pp. 851–853, 2011.

[8] Y. L. Ueng, C. Y. Leong, C. J. Yang, C. C. Cheng, K.-H. Liao, and S. W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I*, vol. 59, pp. 385–398, 2012.