An Evaluation of Design Framework for Min-Sum Irregular LDPC Decoders

Jimpu Suzuki, Hiroshi Tsutsui, and Takeo Ohgane
Graduate School of Information Science and Technology, Hokkaido University

***Abstract*—Design of LDPC decoder depends on its check matrix. Since there exist a lot of check matrices with various sizes, it is not feasible to design a dedicated LDPC decoder for each check matrix. This work aims to make a versatile design framework to generate an LDPC decoder for each given check matrix. We consider a method to reduce the circuit area, focusing on the feature of the check matrix of 5G. In this paper, we present evaluation results of our framework, including gate count evaluations. We evaluated circuit areas of the decoders conforming to 5G. In the case of a check matrix where the number of information bits is about 120, the number of gates is about 3.7 M gates.**

I. Introduction

Recently, LDPC codes [1], one of the error correction codes, have become widely used in wireless communications thanks to its high error correction capability [2] and the ease of parallel processing. Specifically, LDPC codes were standardized as channel code for 10 Gigabit Ethernet, WiMAX [3], and 5G [4]. Since various types and sizes of LDPC check matrices are used depending on the standards, dedicated decoders must be designed to operate along with each given check matrix. Therefore, the cost of optimizing each architecture to meet various standards can be enormous. To solve this problem, we are developing of a framework for LDPC decoders to generate optimal architectures automatically.

This framework enables the automatic generation of LDPC decoders' architecture for any check matrix. The approach is almost the same as one in [5]. In this paper, we propose a method generating irregular LDPC decoders' architecture and show results of circuit area evaluation using 5G check matrices as a demonstration. In addition, we propose a method to reduce the circuit area of the decoders generated by the proposed framework for LDPC codes used in 5G considering its check matrix structure.

This paper is organized as follows. In Sec. II, we briefly review the LDPC codes to explain the proposed approach. In Sec. III, we explain the proposed approach and its LDPC decoder architecture. In this section, we also explain the proposed method to reduce the circuit area for 5G LDPC. In Sec. IV, we evaluate the circuit areas and throughputs of decoders constructed by using the proposed framework under 5G conditions. Finally, Sec. V concludes this paper.

II. LDPC codes

In this section, we describe the sum-product and min-sum algorithms of LDPC codes. Fig. 1 shows a flowchart of the sum-product decoding algorithm. The decoding procedure is divided into row processing, code estimation, and column processing. By repeating these processes, errors are corrected.

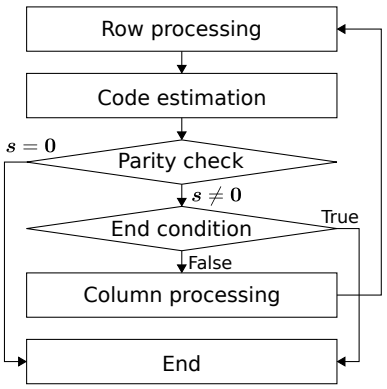


Fig. 1. The flowchart of LDPC decoding

First, for each row and column of the $M \times N$ check matrix H, $A(m)$ and $B(n)$ are defined as

$$A(m) := \{n | H_{mn} = 1\}, \quad (1)$$

$$B(n) := \{m | H_{mn} = 1\}, \quad (2)$$

where $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$. The numbers of elements in $A(m)$ and $B(n)$ are called row weight $w_{\mathrm{r}}(m)$ and column weight $w_{\mathrm{c}}(n)$, respectively. Next, the log-likelihood ratio $\boldsymbol{\lambda} = \{\lambda_n\}$ is calculated from the received words by

$$\lambda_n = \ln \frac{P(y_n | x_n = 0)}{P(y_n | x_n = 1)}, \quad (3)$$

where x_n and y_n are sent bit and received signal, respectively. In particular, for the additive white Gaussian noise (AWGN) channel, $\boldsymbol{\lambda}$ is calculated by

$$\lambda_n = \ln \frac{2y_n}{\sigma^2}, \quad (4)$$

where σ^2 is the variance of noise in the channel. In this paper, we assume that the channel is the AWGN channel. The log-likelihood ratio $\boldsymbol{\lambda}$ is a value used in column processing and code estimation described below.

A. Initialization

For (m, n) satisfying $H_{mn} = 1$, the values of β_{mn} which are used to code estimation are initialized as

$$\beta_{mn} = \lambda_n. \quad (5)$$

Also, the maximum number of iterations is set in the initialization.

B. Row processing

For each row m, the values of α_{mn} corresponding to (m,n) satisfying $H_{mn}=1$ are calculated and updated by

$$\alpha_{mn} = \left(\prod_{\acute{n}\in(A(m)\backslash n)} \mathrm{sign}(\beta_{m\acute{n}})\right) f\left(\sum_{\acute{n}\in(A(m)\backslash n)} f\left(|\beta_{m\acute{n}}|\right)\right), \quad (6)$$

where $f(x)$, $x>0$, is defined by

$$f(x) = \ln\frac{\exp(x)+1}{\exp(x)-1} = -\ln\tanh\left(\frac{x}{2}\right). \quad (7)$$

The values of α_{mn} are used to update the values of β_{mn}. In this paper, we calculate α_{mn} using an approximation [6] as

$$f\left(\sum_{\acute{n}\in(A(m)\backslash n)} f(|\beta_{m\acute{n}}|)\right) \approx f\left(f\left(\min_{\acute{n}\in(A(m)\backslash n)}|\beta_{m\acute{n}}|\right)\right) \approx \min_{\acute{n}\in(A(m)\backslash n)}|\beta_{m\acute{n}}|. \quad (8)$$

Substituting (8) into (6), we obtain

$$\alpha_{mn} = \left(\prod_{\acute{n}\in(A(m)\backslash n)} \mathrm{sign}(\beta_{m\acute{n}})\right) \min_{\acute{n}\in(A(m)\backslash n)}|\beta_{m\acute{n}}|. \quad (9)$$

The decoding algorithm using this approximation is called the min-sum algorithm. Compared to the sum-product decoding algorithm, the min-sum algorithm does not include complex processes such as calculating the hyperbolic tangent functions. The min-sum algorithm is suitable for hardware implementations because of its low complexity. In this paper, we utilize the min-sum algorithm.

C. Code estimation

The values of α_{mn} and $\boldsymbol{\lambda}$ are used to estimate the original code as

$$Q_n = \lambda_n + \sum_{\acute{m}\in\beta(n)} \alpha_{\acute{m}n}, \quad (10)$$

$$\hat{y}_n = \begin{cases} 0 & \text{if } Q_n > 0 \\ 1 & \text{if } Q_n < 0, \end{cases} \quad (11)$$

where $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)^{\mathrm{T}}$ is the estimation. From the estimated code and the check matrix, the syndrome $\boldsymbol{s}$ is calculated by

$$\boldsymbol{s} = H\hat{\boldsymbol{y}}. \quad (12)$$

If $\boldsymbol{s}=\boldsymbol{0}$, the estimated code is a code word, hence the process is terminated, and $\hat{\boldsymbol{y}}$ is output. If $\boldsymbol{s}\neq\boldsymbol{0}$, the process continues. Even if $\boldsymbol{s}\neq\boldsymbol{0}$, when the number of iterations at this point is equal to the preset maximum number of iterations, $\hat{\boldsymbol{y}}$ is output and the process is terminated.

D. Column processing

For each column m, the values of β_{mn} corresponding to (m,n) satisfying $H_{mn}=1$ are culculated and updated by

$$\beta_{mn} = \sum_{\acute{m}\in(B(n)\backslash m)} \alpha_{\acute{m}n} + \lambda_n. \quad (13)$$

When this process is complete, we return to row processing and repeat the same processes.

III. Proposed Method

In this section, we describe the architectural configuration of the irregular LDPC decoders generated by the proposed framework. This framework generates the architecture of the decoder corresponding to a given check matrix as a register-transfer level (RTL) description using Verilog-HDL.

Fig. 2 shows the overall architectural configuration of the decoders generated by the proposed framework, which is similar to one proposed in [5]. The architecture consists of SRAMs storing α and β values, a control unit, and a ROM which contains address values used for the control.

The control unit contains N column processing modules and M row processing modules. Each column processing module performs the corresponding column processing for a column. Column processing modules work in parallel for all columns. Row processing modules work in the same manner.

The SRAMs read and write α or β values to and from the control unit at appropriate timing for row and column processing. SRAMs for α and β consist of N and M banks, respectively, to prevent the memory access conflict. In addition, we have implemented read/write scheduling to avoid the memory access conflict for each bank. This memory access schedule is calculated in advance from the given check matrix H. The scheduling data for writing to SRAM banks is stored in ROM, and the scheduling data for reading from SRAM banks is embedded as combinational circuits in the column and row processing modules.

A. Row processing module

The block diagram of the row processing module is shown in Fig. 3. This module computes α_{mn} values according to (9).

Here, we define the maximum row weight as

$$w_{\mathrm{r,max}} = \max_m w_{\mathrm{r}}(m). \quad (14)$$

The row processing module contains $w_{\mathrm{r,max}}$ circuits to find the smallest absolute value of the input β_{mn} values, and multiply it by the exclusive OR of the sign bits of the input β_{mn} values. The β_{mn} values read from the corresponding banks of SRAM β_m are input to these circuits using $w_{\mathrm{r,max}}$ cycles. The first circuit ignores the first input, the second circuit ignores the second input, and so on, to realize $\acute{n}\in(A(m)\backslash n)$ in (9). At the end of the processes, α_{mn} are output from the $w_{\mathrm{r,max}}$ circuits in reversed order.

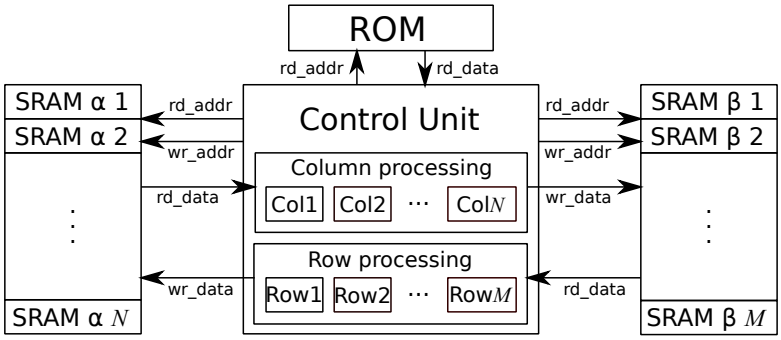


Fig. 2. The architectural configuration of the irregular LDPC decoders generated by the proposed framework

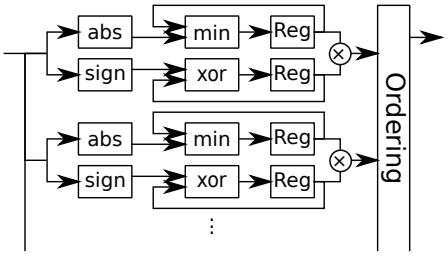


Fig. 3. The block diagram of the proposed row processing module

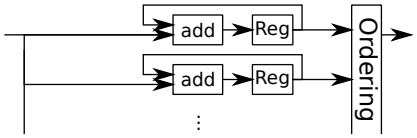


Fig. 4. The block diagram of the proposed column processing module

B. Column processing module

The block diagram of the column processing module is shown in Fig. 4. This module computes the β_{mn} values according to (13).

The data flow is similar to the row processing modules. Let us define the maximum column weight as

$$w_{\mathrm{c,max}} = \max_{m} w_{\mathrm{c}}(n). \tag{15}$$

The column processing module contains $w_{\mathrm{c,max}}$ circuits that accumulate input values. Note that the code estimation calculations are included in these modules.

C. Memory bank

The data allocation to the SRAM banks is performed based on the given check matrix H, whose size is $M \times N$. SRAM α and SRAM β are divided into N and M banks, respectively. The nth bank of SRAM α stores α_{mn} for $m \in B(n)$, while the mth bank of SRAM β stores β_{mn} for $n \in B(m)$. The log-likelihood ratio λ_n is kept at the top of the nth bank of SRAM α. The depths of memory are set to $w_{\mathrm{c,max}}$ and $w_{\mathrm{r,max}}$ for SRAM α and β, respectively, with zero filling if necessary. In addition, in order to prevent conflict of memory accesses, the scheduling of memory accesses is performed in advance and the scheduling data is stored in the ROM. Memory access is controlled by using the scheduling data in this ROM.

D. Circuit area reduction for 5G LDPC

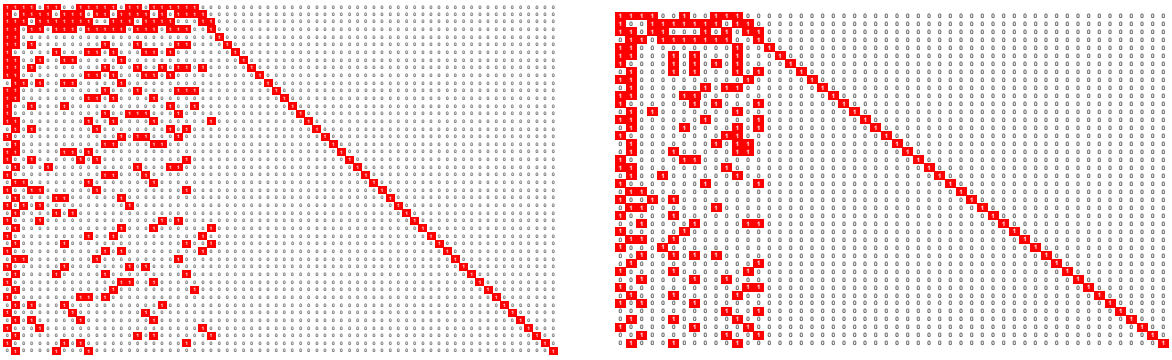
In this paper, we also propose a method to reduce the circuit area of the decoders generated by this framework for LDPC codes used in 5G. Fig. 5 shows the base graphs of the 5G LDPC. Hereafter, we call them BG1 and BG2. From these two base graphs, the check matrices are generated by replacing each element of base graphs with $Z_c \times Z_c$ matrices, where Z_c is a parameter. The zero elements are replaced by zero matrices, while the non-zero elements are replaced by shifted identity matrices whose shift amount varies depending on the position.

The base graphs are mostly occupied by an identity matrix-like structure as we can see from Fig. 5. In this part, the column weight w_{c} equals 1, and the β values are always constant even if the column processing is repeated. Therefore, there is no need for column processing for this part. Based on this idea, we can exclude the column processing circuits for the columns of $w_{\mathrm{c}} = 1$. By reducing the number of column processing circuits, it is expected that the decoder size can be reduced. This method can reduce the number of column processing circuits by $44Z_c$ and $38Z_c$ in the cases of BG1 and BG2, respectively. Because the reduction rate of the number of column processing circuits is approximately 60–70 %, the circuit area derived from column processing circuits may be significantly reduced.

IV. Result and Evaluation

We have synthesized irregular LDPC decoders generated by the proposed framework using 0.13 μm CMOS cell library, and evaluated the circuit area for several different values of the parameter Z_c. In this paper, as an example, a 5G BG2 check matrix was used for demonstration. In this synthesis, an operating frequency was set to 100 MHz. The results are shown in Table I without and with the circuit area reduction for 5G LDPC. All the numbers of gates below do not include SRAMs.

When $Z_c = 20$, information bit length is approximately 120, and the circuit area is 3.7 M gates. If a larger value of Z_c which corresponds to the practical information bit length used in 5G is to be implemented, the circuit area will become too large.



Base graph 1 (BG1) Base graph 2 (BG2)

Fig. 5. Structure of 5G LDPC base graphs. Colored cells are non-zero elements and others are zero elements.

TABLE I
TOTAL GATE NUMBER FOR DIFFERENT Z_c VALUES

Z_c	Gate count without reduction	Gate count with reduction	Reduced gates	Reduction rate (%)
2	380,556	372,459	8,097	2.128
3	569,702	557,549	12,153	2.133
4	760,111	744,547	15,564	2.048
5	959,792	937,631	22,161	2.309
6	1,146,268	1,122,399	23,869	2.082
⋮	⋮	⋮	⋮	⋮
10	1,908,276	1,864,804	43,472	2.278
⋮	⋮	⋮	⋮	⋮
20	3,790,397	3,709,511	80,886	2.134

The circuit areas using the reduction approach considering the base graph structure are smaller than those without the reduction. However, the reduction rate is only about 2 %. This small performance gain is because the logic synthesis automatically optimized the decoders without the reduction approach.

The output throughput T of the LDPC decoders is defined by

$$T = \frac{F_{\max} M}{(N_{\mathrm{c}} + 2N_{\mathrm{w}})I} \quad \text{bit/sec} \tag{16}$$

where M is the information bit length, I is the maximum number of iterations, N_{c} is the total number of the row/column processing cycles for each iteration, N_{w} is the wait cycles for row/column processing due to the gap between the row/column weights and its maximum, and $F_{\max}$ is the operating frequency. The longer the bit length of information bits is dealt with the higher the throughput is.

In the case of BG2 implementations, $N_c = 3w_{\mathrm{c,max}} + w_{\mathrm{r,max}} + 3 = 82$, $N_w = 0$, and the maximum information bit length is $M = 3,840$. The maximum throughput can be 468 Mbps when $I = 10$.

V. CONCLUSIONS

In this paper, we proposed a framework to generate LDPC decoders. Also, considering the structure of the check matrices of 5G LDPC codes, we proposed a circuit area reduction method. The proposed method is expected to relieve hardware design difficulties for various check matrices including irregular LDPC codes.

Using 5G LDPC check matrices, we demonstrated the effectiveness of the proposed framework. However, the circuit area of the decoders generated by the proposed framework exceeds a few M gates in the case of a check matrix whose number of information bits is about 120 used in 5G. If the code length is longer than that demonstrated in this paper, the area of the circuit is expected to be too large for implementation. The reduction rate of the circuit area utilizing the proposed approach is not as large as expected and the proposed method does not give a significant performance gain. According to the comparison, the reduction rate was about 2 %. In the future, we would like to reduce the circuit area based on the proposed framework and develop architectures of irregular LDPC decoders that can be implemented on a practical scale.

ACKNOWLEDGEMENT

The authors would like to thank the GI-CoRE GSB, Hokkaido University for fruitful discussions. This work is supported in parts by the Ministry of Internal Affairs and Communications for SCOPE Program (185001003). This work is also supported through the activities of VDEC, the University of Tokyo, in collaboration with Synopsys, Inc., Cadence Design Systems, Inc., and Mentor Graphics, Inc.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[3] Wimax Forum, http://www.wimaxforum.org/.
[4] 3rd Generation Partnership Project, "Technical specification group radio access network; NR; Multiplexing and channel coding (release 15)," in *3GPP TS 38.212 V15.0.0*, Dec. 2017.
[5] J. Broulím, P. Broulím, J. Moldaschl, V. Georgiev, and R. Šalom, "Fully parallel FPGA decoder for irregular LDPC codes," in *Proc. Telecommunications Forum (TELFOR)*, Nov. 2015, pp. 309–312.
[6] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673–680, May 1999.