An LDPC Decoder with SNR Information

Kai-Jiun Yang, Shang-Ho Tsai, and Heng-Chang Hsu
Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan

***Abstract*—In this work an LDPC decoder which complies with IEEE 802.11n is proposed and implemented. The code rate is $1/2$ and the code length is 648. We used partially parallel structure to reduce the area. Additionally the SNR information is applied to improve the BER performance. Moreover the CNU and the BNU in the min-sum-correct algorithm were reordered so that the hardware complexity can be reduced, and early termination can be achieved at the first iteration. Furthermore the parity check matrix is reordered such that the latency of each iteration is reduced by $1/3$. The proposed LDPC decoder can reach a throughput of $37 \sim 319$Mbps with a core area of $5.3mm^2$ and power consumption 224mW in a TSMC 90nm process.**

I. Introduction

As the VLSI process technology advances, the low-density parity-check (LDPC) code proposed by Gallager [1] caught more and more attention. Chung *et al.* developed an implementation that approaches the Shannon limit within 0.0045 dB performance loss for binary-input AWGN channel [2]. Since the computing complexity is lower than Turbo codes and the error-correct capability outperforms, the use of LDPC codes has been included in several communication standards such as IEEE 802.11n Wi-Fi standard, IEEE 806.16e WiMAX, and IEEE 802.3an LAN protocols.

The hardware implementations of the LDPC decoders were daunting due to the large parity check matrix. In general there are two types of the LDPC decoder: fully and partially parallel forms. Fully parallel form maps the entire Tanner graph into the hardware. As a result fully parallel LDPC decoder can achieve very high throughput but demand huge hardware. Blanksby and Howland implemented the first LDPC decoder chip in fully parallel form with a maximum throughput of 1Gb/s at 64MHz clock frequency [3]. In order to achieve parallelism with 64 decoding iterations, the functional blocks were repeatedly duplicated. The chip dimension is $52.5mm^2$ using a $0.16\mu m$ process.

On the other hand, partially parallel form shares the calculation units to cut down the area. Consequently the throughput is lower and the routing is congested. Shih *et al.* proposed an code rate 1/2 LDPC decoder which complies with IEEE 802.16e standard [4]. The throughput varied from 60.6 Mb/s with 8 iterations to maximum 222.2Mb/s with 2 iterations at 83.3MHz clock frequency, and the die size is only $8.29mm^2$ in a $0.13\mu m$ process technology.

In this paper, we find that the SNR information helps to boost BER performance in the min-sum-correct algorithm with few iterations. The simulation showed that the performance can be improved by 0.3 dB at BER $= 10^{-4}$ with 8 iterations. Furthermore we propose to reorder the check-node-update (CNU) and the bit-node-update (BNU) in the min-sum-correct algorithm to minimize the routing complexity and the computing iteration. Moreover the reordered CNU and BNU can check whether the decoded codeword is correct at the first iteration so as to perform early termination. Also each iteration can be shorten if CNU and BNU work concurrently if there is no dependency. The proposed LDPC decoder was implemented in a TSMC 90nm process with die size $5.3mm^2$ and it can be operated at a maximum clock frequency of 151MHz. The throughput ranges from 39 Mb/s with 16 iterations to 319 Mb/s with 1 iteration.

II. Low Density Parity Check Codes

LDPC code is a systematic block code. The $M \times N$ parity check matrix $\mathbf{H}$ contains lots of 0s and few 1s. The LDPC code has N information bits and $M - N$ parity bits such that the code rate R is N/M. IEEE 802.11n and IEEE 802.16e standards use quasi-cyclic (QC) LDPC codes. Fig. 1 shows an example of a QC LDPC parity check matrix with code length 648 bits and information length 324 bits.In this figure, each number or dash in the matrix is the base matrix with dimension $Z \times Z$, where Z can be 27, 54, or 81. The base matrix denoted by "-" is a zero matrix and the value 0 stands for identity matrix. Other values indicates the amount of cyclic right-shift of the identity matrix.

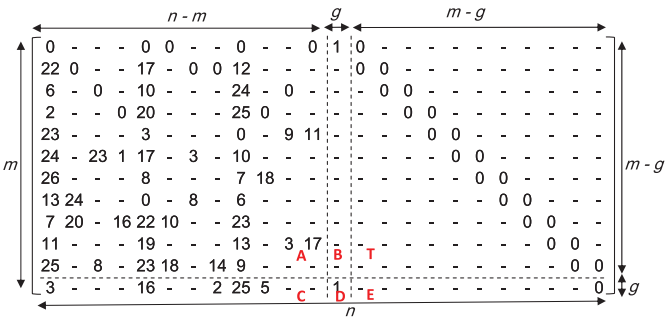

Fig. 1. An example of quasi-cyclic LDPC code parity check matrix $\mathbf{H}$ used in 802.11n with $Z = 27$.

A. LDPC Encoder

Let $\mathbf{u}$ be the information vector, $\mathbf{G}$ be the LDPC generator matrix, and $\mathbf{v}$ be the generated codeword vector such that $\mathbf{uG} = \mathbf{v}$. Since LDPC code is a linear block code, the parity check matrix $\mathbf{H}$ satisfies $\mathbf{Hv}^T = 0$. The equations can be reformed as

$$\mathbf{Hv}^T = \mathbf{H}(\mathbf{uG})^T = \mathbf{HG}^T\mathbf{u}^T = 0 \Longrightarrow \mathbf{HG}^T = 0 \qquad (1)$$

Richardson and Urbanke suggested an efficient method to generate $\mathbf{G}$ from the stored $\mathbf{H}$ to decrease the use of storage

978-1-4799-0434-1/13/$31.00 ©2013 IEEE ICICS 2013

Authorized licensed use limited to: National Chiao Tung Univ.. Downloaded on June 09,2022 at 08:47:54 UTC from IEEE Xplore. Restrictions apply.

elements [5]: Let the parity check matrix $\mathbf{H}$ be partitioned into six sub-matrices $\mathbf{A}_{(m-g)\times(n-m)}$, $\mathbf{B}_{(m-g)\times g}$, $\mathbf{C}_{g\times(n-m)}$, $\mathbf{D}_{g\times g}$, $\mathbf{E}_{g\times(m-g)}$, and $\mathbf{T}_{(m-g)\times(m-g)}$ as illustrated in Fig.1. Also the codeword $\mathbf{v}$ is segmented as $\mathbf{v}=[\mathbf{u}\ \mathbf{p}_1\ \mathbf{p}_2]$, where $\mathbf{u}$ is the information bits with length $n-m$ and the parity check bits $\mathbf{p}_1$ and $\mathbf{p}_2$ are of length g and and $m-g$ respectively. Therefore $\mathbf{H}\mathbf{v}^T=0$ becomes

$$\begin{cases} \mathbf{A}\mathbf{u}^T+\mathbf{B}\mathbf{p_1}^T+\mathbf{T}\mathbf{p_2}^T=0 \\ \mathbf{C}\mathbf{u}^T+\mathbf{D}\mathbf{p_1}^T+\mathbf{E}\mathbf{p_2}^T=0 \end{cases} \tag{2}$$

From Eqs. (2), we have $(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C})\mathbf{u}^T+(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D})\mathbf{p}_1^T=0$. Let $\phi:=-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}$. With appropriate combination of $\mathbf{B}$, $\mathbf{T}$, $\mathbf{D}$, and $\mathbf{E}$, the matrix ϕ becomes an identity matrix. Then we can obtain the mathematical relationship as follows:

$$\begin{cases} \mathbf{p}_1^T=(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C})\mathbf{u}^T \\ \mathbf{p}_2^T=\mathbf{T}^{-1}(\mathbf{A}\mathbf{u}^T+\mathbf{B}\mathbf{p}_1^T) \end{cases} \tag{3}$$

As a result, the encoding process can be broken down as shown in Fig.2. The information is encoded as codeword $\mathbf{v}$ and then modulated as $\mathbf{x}$ by BPSK. Afterwards it is sent into AWGN channel with noise variance σ^2 and received as $\mathbf{y}$.

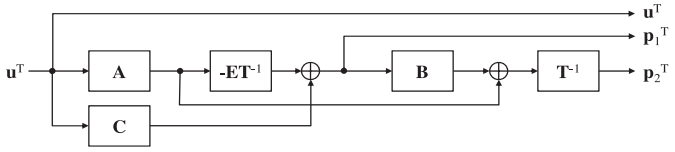

Fig. 2. Block diagram of the LDPC code encoder.

B. LDPC Decoder

It is common to use sum-product algorithm [6] for decoding, which iterates the soft messages between bit-nodes and check-nodes in the Tanner graph for best approximating the source information as drawn in Fig.3. A log-likelihood ratio (LLR) is usually applied to trim the complexity.

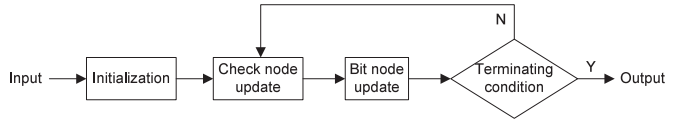

Fig. 3. Decoding flow of an LDPC decode.

1) Initialization: Each bit-node B_i is initialized according to the LLR from the received y_i

$$L_{B_i}=\log\frac{P(v_i=0)}{P(v_i=1)}=\log(e^{\frac{-2y_i}{\sigma^2}})=\frac{-2y_i}{\sigma^2} \tag{4}$$

2) Check-node update (CNU): The LLR of the path from C_j to B_i is acquired from the connected bit-nodes except from B_i as shown in Fig.4(a).

$$\begin{aligned} L_{r_{j\to i}} &= \log\frac{P(r_{j\to i}=0)}{P(r_{j\to i}=1)} \\ &= \log\frac{1+\prod_{i'\in W(j)\setminus\{i\}}(1-2P(q_{i'\to j}))}{1-\prod_{i'\in W(j)\setminus\{i\}}(1-2P(q_{i'\to j}))}. \end{aligned} \tag{5}$$

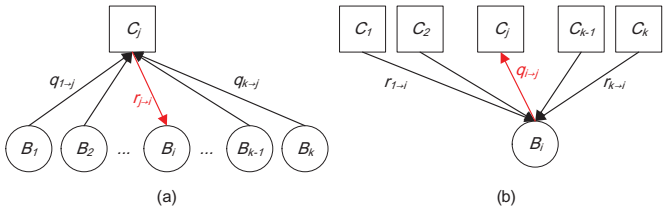

Fig. 4. The message passing relationship (a) check-node update, and (b) bit-node update.

where $r_{j\to i}$ is the probability message from check-node j to bit-node i. The min-sum-correct algorithm was proposed in [7] for achieving a good approximation. The fundamental CNU includes three bit-nodes and one check-node. Let the LLRs tethered to these nodes be $L_{q_{1\to1}}$, $L_{q_{2\to1}}$, and $L_{r_{1\to3}}$. By using the Jacobian logarithm twice, Eq. (5) can be reformed as

$$\begin{aligned} L_{r_{1\to3}} &= \log\frac{1+(\frac{e^{L_{q_{1\to1}}}-1}{e^{L_{q_{1\to1}}}+1}\cdot\frac{e^{L_{q_{2\to1}}}-1}{e^{L_{q_{2\to1}}}+1})}{1-(\frac{e^{L_{q_{1\to1}}}-1}{e^{L_{q_{1\to1}}}+1}\cdot\frac{e^{L_{q_{2\to1}}}-1}{e^{L_{q_{2\to1}}}+1})} \\ &= \max(0,L_{q_{1\to1}}+L_{q_{2\to1}})+\log(1+e^{-|L_{q_{1\to1}}+L_{q_{2\to1}}|}) \\ &\quad -\max(L_{q_{1\to1}},L_{q_{2\to1}})-\log(1+e^{-|L_{q_{1\to1}}-L_{q_{2\to1}}|}) \\ &= \mathrm{sign}(L_{q_{1\to1}})\,\mathrm{sign}(L_{q_{2\to1}})\min(|L_{q_{1\to1}}|,|L_{q_{2\to1}}|) \\ &\quad +\log(1+e^{-|L_{q_{1\to1}}+L_{q_{2\to1}}|})-\log(1+e^{-|L_{q_{1\to1}}-L_{q_{2\to1}}|})). \end{aligned} \tag{6}$$

3) Bit-node update (BNU): Similarly, the LLR of the path from B_i to C_j is acquired from the connected check-nodes except from C_j as shown in Fig.4(b).

$$\begin{aligned} L_{q_{i\to j}} &= \log\frac{P(v_i=0)\prod_{j'\in M(i)\setminus\{j\}}P(r_{j\to i'}=0)}{P(v_i=1)\prod_{j'\in M(i)\setminus\{j\}}P(r_{j\to i'}=1)} \\ &= L_{B_i}+\sum_{j'\in M(i)\setminus\{j\}}L_{r_{j'\to i}}. \end{aligned} \tag{7}$$

4) Terminating condition: The updated LLR of bit-node B_i is given by

$$L_{B_i}^{post}=\log\frac{P^{post}(v_i=0)}{P^{post}(v_i=1)}=L_{B_i}+\sum_{j\in M(i)}L_{r_{j'\to i}}. \tag{8}$$

The estimated v_i is 1 when the posteriori probability $P^{post}(v_i=1)>P^{post}(v_i=0)$ i.e.

$$\begin{cases} v_i\Rightarrow 0, & \text{if } L_{v_i}^{post}\geq 0 \\ v_i\Rightarrow 1, & \text{if } L_{v_i}^{post}<0 \end{cases} \quad \forall i\in\{1,2,\ldots,n\}. \tag{9}$$

If the codeword is correct, the iteration stops. Otherwise the iteration continues until the predefined iteration limit.

III. Proposed Algorithm and Architecture

This work proposes to utilize the known SNR information to further enhance the BER performance. In addition the CNU and BNU in min-sum-correct algorithm are reordered for early termination so as to speed up the lengthy decoding whenever the channel quality allows. Based on the proposed features the

Authorized licensed use limited to: National Chiao Tung Univ.. Downloaded on June 09,2022 at 08:47:54 UTC from IEEE Xplore. Restrictions apply.

block diagram of the proposed LDPC decoder is as shown in Fig.5. The proposed architecture is in partial parallel form so it is suitable for mobile devices, where chip-size and power-consumption are usually more important than throughput.

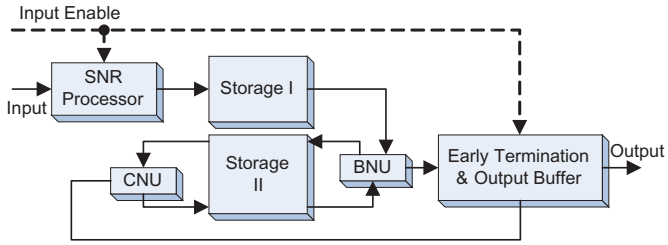

Fig. 5. The architecture of the proposed LDPC decoder. The dash line is the control signal.

.

A. SNR Processor

The SNR processor multiplies the input y_i with the known SNR value. Usually the SNR information can be acquired from comparing the pilot tones with pre-defined patterns. At the initialization stage of LDPC decoding, the LLR of bit-nodes are initialize by Eq. (4), which has the inverse term of the variance. According to the definition of SNR, it is proportional to the inverse of the deviation σ. Therefore, multiplying the SNR to the input y_i can best initialize the bit-nodes.

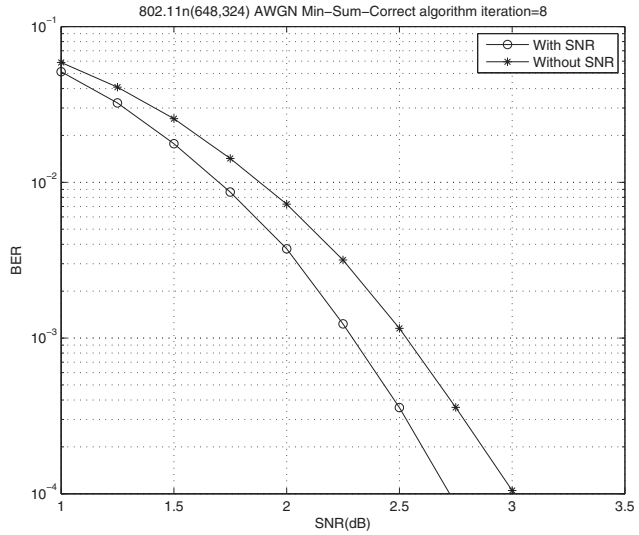

Fig. 6. The BER of 8-iteration LDPC decoding with and without SNR information.

An example is provided to show that knowing the SNR information can indeed improve the decoding performance. Fig. 6 is the BER performance of a 8-iteration LDPC decoding with and without SNR information. At BER=10^{-4}, the SNR improves by 0.3dB if the SNR information is applied. On the other hand, we have simulated off-line and found that the improvement starts to decrease as the number of iteration increases or the number of iterations equals to one. Therefore, the number of iterations can be decreased on the premise of utilizing SNR information. The multiplication of the SNR information is implemented by table look-up to reduce complexity, and this is shown in Tab. I.

TABLE I
THE SNR MAPPING TABLE.

$SNR_{integer}$	Select	$SNR_{integer}$	Select
[$-\infty$,2)	1	[2,3)	2
[3,4)	3	[4,5)	4
[5,6)	5	[6,7)	6
[7,8)	7	[8,9)	8
[9,10)	9	[10,∞)	10
$SNR_{decimal}$	Select	$SNR_{decimal}$	Select
[0,0.25)	0	[0.25,0.5)	0.25
[0.5,0.75)	0.5	[0.75,1)	0.75

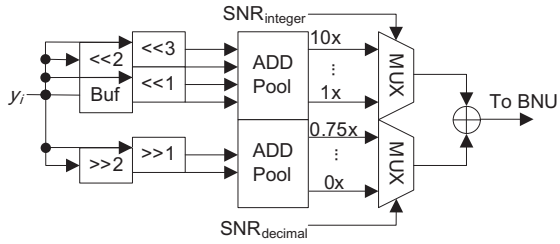

Fig. 7. The schematic of the proposed SNR processer.

Adders and shift-registers are applied to replace the multiplier as shown in Fig. 7. Based on the simulation results, the proposed SNR processor can pronouncedly improve the performance when the SNR is in the range between $0 \sim 10$ dB. Therefore, the SNR mapping table in Tab. I is designed according to this simulation results.

B. Reordering of CNU and BNU

When the system is with high SNR, the received codeword may already be correct without any correction. In this case, there is no needed to perform any iteration in the LPDC decoding procedure. However, for conventional implementation, although early termination can be applied, iterations are still needed to verify whether the received codeword is correct or not. That is, from Fig. 3, BNU and CNU still needed to be conducted once to obtain the initial hard-decision result. To overcome this issue, we propose to reorder the CNU and BNU as shown in Fig. 8 so that the decoded results can be generated sooner at high SNR. At bit-node update, the hard-decision codeword can be acquired and the initialization can be performed concurrently. If the weighting is higher than the threshold, there is no need to continue with check-node update. It is worth to emphasize that the proposed reordering does not affect the performance. It only modifies the decoding flow for better efficiency. As demonstrated in Tab.II, the proposed reordering can generate the output at least two steps earlier.

The proposed reordering also increases hardware reusability. Originally the soft information needs to be registered during initialization and expanded in CNU These different steps

Authorized licensed use limited to: National Chiao Tung Univ.. Downloaded on June 09,2022 at 08:47:54 UTC from IEEE Xplore. Restrictions apply.

Step	1	2	3	4	5	6	7	8
Original algorithm	Initialize	CNU	BNU	CNU	BNU	CNU	BNU	CNU
Hard-decision vlue generate			1st		2nd		3rd	
Early termination compare						1st		2nd
Reordered algorithm	BNU	CNU	BNU	CNU	BNU	CNU	BNU	CNU
Hard-decision vlue generate	1st		2nd		3rd		4th	
Early termination compare				1st		2nd		3rd

TABLE II
COMPARISON OF ORIGINAL ALGORITHM AND THE PROPOSED REORDERED ALGORITHM.

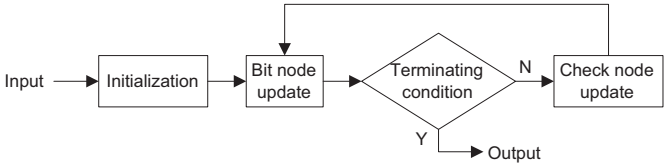

Fig. 8. The reordered min-sum-correct algorithm.

	BNU1								BNU2								BNU3							
C	0	-	1	0	-	-	-	-	0	0	0	-	-	-	-	0	-	-	-	-	-	-	-	-
N	10	20	-	-	0	0	-	-	7	22	23	-	16	-	-	-	-	-	-	-	-	-	-	-
U	-	-	-	-	-	0	0	-	11	19	13	-	-	-	3	17	-	-	-	-	-	-	-	-
1	18	-	-	-	-	0	0		25	23	9	8	-	14	-	-	-	-	-	-	-	-	-	-
C	-	-	1	-	-	-	-	0	3	16	25	-	-	2	-	-	-	5	-	-	-	-	-	-
N	-	24	-	-	0	-	-	-	13	0	6	-	-	-	-	-	8	-	-	-	-	-	-	0
U	-	-	0	-	-	-	-	-	25	8	7	-	-	-	-	-	-	18	-	-	-	-	0	0
2	-	0	-	0	-	-	-	-	22	17	12	-	-	0	-	-	0	-	0	-	-	-	-	-
C	-	-	-	-	-	-	-	-	23	3	0	-	-	-	9	11	-	-	-	-	0	0	-	-
N	-	-	-	-	-	-	-	-	2	20	25	-	0	-	-	-	-	0	-	0	0	-	-	-
U	-	-	-	-	-	-	-	-	6	10	24	0	-	-	0	-	-	-	0	0	-	-	-	-
3	-	-	-	-	-	-	-	-	24	17	10	23	1	-	-	-	3	-	-	-	-	0	0	-

Fig. 9. The proposed parity check matrix for IEEE 802.11n LDPC code.

requires two sets of routing. In the proposed reordering since the initialization can be done concurrently with BNU, the same routing can be shared by the following CNU. Therefore the routing- congestion can be relieved.

C. Latency Reduction

We propose to reorder the parity check matrix such that each overlong iteration can be shorten. Both the storage size and the decoding time increase as the parity check matrix of LDPC codes grow especially using partial parallel method. If the parity check matrix is properly reordered, the decoding latency can be reduced because some CNU and BNU can be operated at the same time (see [8], [9]). From Eq. (1), changing the row-order the product is still zero while changing the column-order only affects the codeword sequence of $\mathbf{v}$. Therefore reordering both the rows and columns of the parity check matrix as shown in Fig.9 does not alter the function. The CNU operates on row data while BNU operates on column data. Furthermore, the reordered parity check matrix is partitioned into three CNU groups and BNU groups. It can be observed that BNU1 does not overlap with CNU3 and BNU3 does not overlap with CNU1. This means that each iteration can overlap by concurrently enabling BNU and CNU over independent groups of data. The iterations can be scheduled as shown in Fig. 10. The proposed latency reduction can save $1/3$ clocks of each iteration.

D. Fixed Point Analysis

Before VLSI implementation the proposed algorithm were simulated with fix-point model for analyzing the quantization noise. The fix-point variables within the design are of 4 integer-bits and 4 decimal-bits. The BER comparison is shown in Fig. 11, where the dashed curves are the floating-point with SNR information, solid curves are the fixed-point with SNR information, and dotted curves are the fixed point without SNR information. It is observed that the BER performance of fixed-point with SNR information is better than that without SNR information.

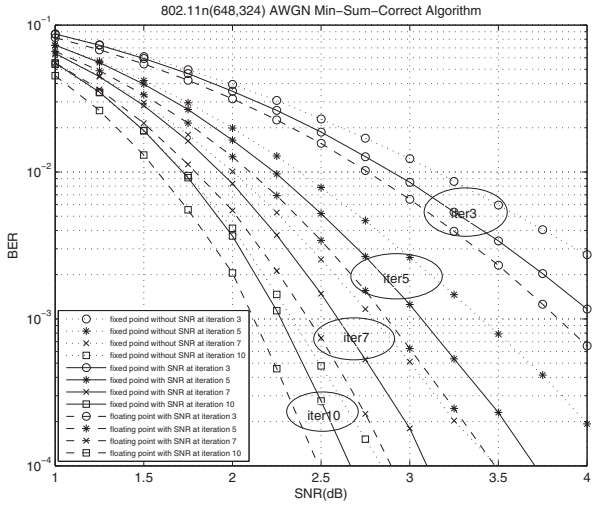

Fig. 11. The BER comparison of floating-point and fixed-point models.

IV. VLSI IMPLEMENTATION

The proposed LDPC decoder for IEEE 802.11n uses a TSMC 90 nm standard cell library for implementation. The layout is as shown in Fig. 12, and the specification is listed in Tab.III.

APR is the most challenging stage during the implementation. Since partial parallel architecture reuses the functional blocks, different check-nodes and bit-nodes need flexible interconnection to share common components. During circuit

Authorized licensed use limited to: National Chiao Tung Univ.. Downloaded on June 09,2022 at 08:47:54 UTC from IEEE Xplore. Restrictions apply.

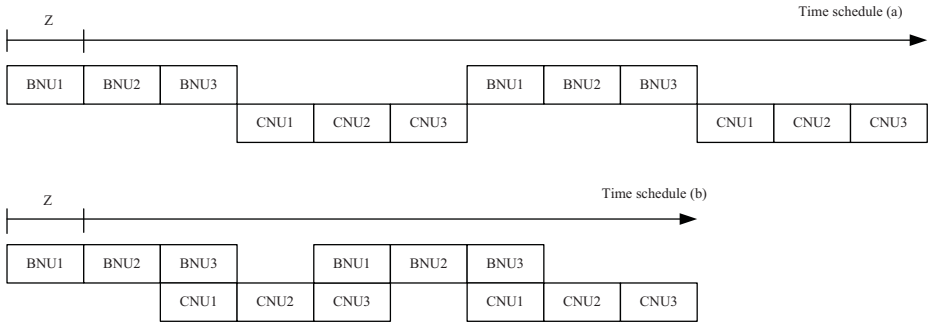

Fig. 10. The timing diagram: (a) Original (b) Proposed. BNU3 with CNU1 can work concurrently, and so do BNU1 with CNU3.

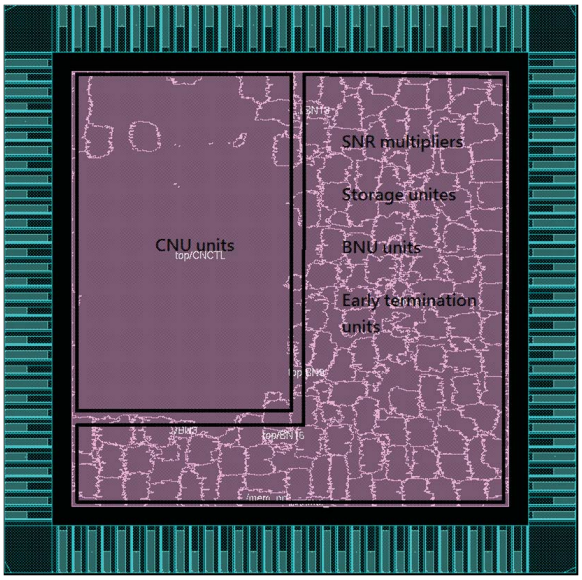

Fig. 12. The layout of the proposed LDPC decoder.

Items	Soecification
Technology	TSMC 90 nm
Package	CQFP128
Chip Size	$5.3mm^2$
Core Area	$3.1mm^2$
Gate Count	674K
Max Frequency	151MHz
Throughput	$37 \sim 319$Mbps
Power Consumption	244mW

TABLE III
THE SPECIFICATION THE PROPOSED LDPC DECODER.

synthesis using Synopsys Design Compiler, the hierarchy must be flattened such that the CAD tool can have the flexibility to perform logic optimization to meet both area and timing constraints. Therefore in the chip layout-view the hierarchy cannot be clearly discerned. On the other hand, it is because the partial parallel architecture is applied, such that the total area can be far less than that of fully parallel architecture at the cost of serial processing time. Additionally the power consumption is reduced due to less chip dimension. With less computing iteration at high SNR, the power saving can be more pronounced.

V. CONCLUSION

We proposed an LDPC decoder compliant with IEEE 802.11n standard, which refers to the known SNR value. With SNR information the BER performance can be improved with fewer number of iterations. Partial parallel architecture is applied in the min-sum-correct algorithm in order to reuse the computing elements, and as a result the area is greatly reduced. Moreover we propose to reorder the CNU and BNU as well as the parity check matrix. By the proposed methods, not only the hardware complexity and computing latency are reduced, but also simpler early-termination can be achieved at the first iteration in LDPC decoding. The proposed LDPC decoder can achieve $37 \sim$319Mbps throughput with $5.3mm^2$ chip area, and the estimated average power consumption is 244mW.

REFERENCES

[1] R. G. Gallager, "Low-Density Parity-Check Codes," Cambridge, MA:MIT Press, 1960.
[2] S. Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit," Communications Letters, IEEE, vol.5, no.2, pp. 58-60, Feb. 2001.
[3] A.J. Blanksby and C.J. Howland, "A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder," Solid-State Circuits, IEEE Journal of , vol.37, no.3, pp.404-412, Mar 2002.
[4] X.Y. Shih, C.Z. Zhan, C.H. Lin, and A.Y. Wu, "An $8.29mm^2$ 52mW Multi-Mode LDPC Decoder Design for Mobile WiMAX System in $0.13\mu m$ CMOS Process," Solid-State Circuits, IEEE Journal of , vol.43, no.3, pp.672-683, March 2008.
[5] T.J. Richardson and R.L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," Information Theory, IEEE Transactions on, vol.47, no.2, pp.638-656, Feb 2001.
[6] S. Lin and D. J. Costello, *Control Coding: Fundamentals and Applications*, 2nd ed. New York: Pearson/Prentice Hall, 2004.
[7] X.Y. Hu, E. Eleftheriou, D.-M. Arnold and A. Dholakia, "Efficient Implementations of the Sum-Product Algorithm for Decoding LDPC Codes," Global Telecommunications Conference, 2001. IEEE , vol.2, pp.1036-1036, 2001.
[8] H. Zhong and T. Zhang, "Design of VLSI Implementation-Oriented LDPC codes," Vehicular Technology Conference, 2003. IEEE 58th , vol.1, pp.670-673, 6-9 Oct. 2003.
[9] I. C. Park and S. H. Kang, "Scheduling Algorithm for Partially Parallel Architecture of LDPC Decoder by Matrix Permutation, in Proc. IEEE ISCAS, pp.5778-5781, May 2005.

Authorized licensed use limited to: National Chiao Tung Univ.. Downloaded on June 09,2022 at 08:47:54 UTC from IEEE Xplore. Restrictions apply.